1992-09

Intelligent information retrieval for a multimedia database using captions.

Guglielmo, Eugene J.

Monterey, California. Naval Postgraduate School

Unclassified

Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification UNCLASSIFIED	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report
2b Declassification/Downgrading Schedule	Approved for public release; distribution is unlimited.
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School / 6b Office Symbol *(If Applicable)* CS	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding/Sponsoring Organization / 8b Office Symbol *(If Applicable)*	9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*	10 Source of Funding Numbers: Program Element Number / Project No / Task No / Work Unit Accession No

11 Title *(Include Security Classification)*

INTELLIGENT INFORMATION RETRIEVAL FOR A MULTIMEDIA DATABASE USING CAPTIONS

12 Personal Author(s)
Guglielmo, Eugene J.

13a Type of Report	13b Time Covered	14 Date of Report *(year, month,day)*	15 Page Count
Ph.D. Dissertation	From Dec 90 To Jul 92	1992, Jul, 23	369

16 Supplementary Notation

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	Information Retrieval, Multimedia Database, Natural Language Processing, Artificial Intelligence Distributed Processing, Type Hierarchy, Prolog, Captions.

19 Abstract *(continue on reverse if necessary and identify by block number*

This report describes an intelligent information retrieval system, MARIE, that employs natural language processing techniques for indexing and retrieving multimedia data. Captions describe photographs from the Naval Air Warfare Center Weapons Division, China Lake, California; the captions were written in English consisting mostly of noun phrases. For our work, an object-oriented type hierarchy represents semantic knowledge. Captions are parsed to produce a logical form, from which nouns and verbs are extracted to form keyword files. User queries are also specified in natural language. A two-phase match process is employed between the query and database. A coarse-grain match searches the keyword files and issues SQL queries to a relational database as necessary to find candidate captions for further analysis. A fine-grain match then compares the logical form of the query to the logical form for each caption. A list of caption IDs and accompanying match scores is then presented to the user, who can view the image and supporting data. A companion technical report contains the programs for the system discussed here.

20 Distribution/Availability of Abstract [X] unclassified/unlimited [ ] same as report [ ] DTIC users	21 Abstract Security Classification UNCLASSIFIED	
22a Name of Responsible Individual Prof. Neil C. Rowe	22b Telephone *(Include Area code)* (408) 646-2462	22c Office Symbol Code CSRp

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted — security classification of this page

All other editions are obsolete — Unclassified

Approved for public release; distribution is unlimited.

Intelligent Information Retrieval for a Multimedia Database Using Captions

by

Eugene J. Guglielmo
Naval Air Warfare Center Weapons Division
B.S., St. John's University, 1979
M.S., California State University, Chico, 1987

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1992

ABSTRACT

This report describes an intelligent information retrieval system, MARIE, that employs natural language processing techniques for indexing and retrieving multimedia data. Captions describe photographs from the Naval Air Warfare Center Weapons Division, China Lake, California; the captions were written in English consisting mostly of noun phrases. For our work, an object-oriented type hierarchy represents semantic knowledge. Captions are parsed to produce a logical form, from which nouns and verbs are extracted to form keyword files. User queries are also specified in natural language. A two-phase match process is employed between the query and database. A coarse-grain match searches the keyword files and issues SQL queries to a relational database as necessary to find candidate captions for further analysis. A fine-grain match then compares the logical form of the query to the logical form for each caption. A list of caption IDs and accompanying match scores is then presented to the user, who can view the image and supporting data. A companion technical report contains the programs for the system discussed here.

3/6712

TABLE OF CONTENTS

I. INTRODUCTION 1
A. INTELLIGENT IR AND MULTIMEDIA DB 1
B. PROBLEM DEFINITION AND BASIC ASSUMPTIONS 3
C. REPORT ORGANIZATION 4

II. SURVEY OF PREVIOUS WORK 5
A. INTRODUCTION 5
B. MULTIMEDIA DATA MODEL APPROACHES 5
C. ENHANCED KEYWORD APPROACHES 7
D. NL APPROACHES 8
E. PARALLEL MATCH APPROACHES 12
F. IMAGE ANALYSIS APPROACHES 13
G. OTHER APPROACHES 14
H. SUMMARY 14

III. SYSTEM APPROACH 16
A. INTRODUCTION 16
B. DATA FILES 17
1. Relational Database 18
2. Semantic Database 18
3. Data Store 20
C. PROCESSES 20
1. NLP Subsystem 20
2. Coarse-Grain Match 21
3. Fine-Grain Match 22
4. Database Update Routines 22
5. User Interface Routines 22
D. A SAMPLE RUN 22
E. SUMMARY 26

IV. THE NLP SUBSYSTEM 27
A. INTRODUCTION 27
B. THE LEXICON 29
1. Lexicon Structure 29

2. Semantic Categories ... 29

C. SYNTACTIC PARSING ... 31

1. Additions ... 32

a. Noun Phrases ... 32

b. Commas ... 32

c. Coordinates ... 34

d. Dates ... 34

e. Ordered Sequence ... 35

2. Complications ... 35

a. Objects of Prepositions ... 35

b. Commas ... 36

c. Embedded Sentences ... 38

d. Conjunctive Noun Phrases ... 40

e. Conjunctive Verb Phrases ... 45

f. Prepositional Phrase Referencing ... 47

g. Dangling Preposition ... 47

h. Nouns with Numbers ... 48

i. Verbals vs Verbs ... 49

j. Case Information for Verbs ... 53

D. FUNCTIONAL PARSING ... 55

1. Functional Parse Categories ... 56

a. General Object (GOBJ) ... 56

b. Prepositional Object (POBJ) ... 57

c. Appositive Object (AOBJ) ... 57

d. ID Object (IOBJ) ... 57

e. Number Objects (NOBJ) ... 57

f. Coordinates (COORD) ... 58

g. Dates (DATE) ... 58

h. General Acts (GACT) ... 58

i. Main Verbs (MAIN-V) ... 58

j. Noun Adjectives (ADJS-NOUN) ... 59

k. Number Adjectives (ADJS-NUM) ... 59

l. Quantifier Adjectives (ADJS-QUANT) ... 59

m. Adjectives (ADJS) ... 59

n. Adverbs (ADV) ... 59
o. Prepositions (PREP) ... 60
p. Number (NUM) ... 60
q. Implicit Verb Case Information ... 60
r. Explicit Verb Case Information ... 61
2. Complications ... 61
a. Sequence of Prepositional Phrases ... 61
b. Conjunctive Nouns ... 64
c. Sentences/Phrases connected using "with" ... 66
d. Adjectives as Verbs ... 68
e. Prepositional Inverses ... 69
E. THE TYPE HIERARCHY ... 70
1. Design Philosophy ... 70
2. Structure ... 72
F. CREATING THE LOGICAL FORM ... 73
1. Logical Form Structure ... 74
2. Noun Phrase Analysis ... 75
a. Specialization ... 76
b. Correlations ... 77
c. Inferring Ownership ... 78
d. Inferring Themes ... 80
e. Inferring Agents ... 81
f. Inferring Isa's ... 81
g. Inferring States ... 82
h. Default Case ... 83
i. Coordinates ... 84
j. Dates ... 85
3. Anaphoric Reference ... 85
4. Inferring Themes outside of Noun Phrases ... 93
5. Inferring Verbs ... 94
6. Quantifiers ... 95
7. Relationship Chains ... 96
G. POSTPROCESSING OF THE LOGICAL FORM ... 97
1. Transformations ... 98
2. Inferring Additional Facts ... 100

H. SUMMARY 102

V. CAPTION MATCHING 104

A. INTRODUCTION 104

B. KEYPHRASE MATCHING 104

C. COARSE-GRAIN MATCHING 106

1. Semantic Database - Keyword Index Files 106

2. Determining the Match Criteria 107

a. Treatment of Verbs 107

b. Treatment of Nouns 108

c. Defining the Function Φ 108

3. Matching using the Semantic Data 109

4. Matching using the Relational Data 110

D. FINE-GRAIN MATCHING 111

1. Instance Matching 114

2. Direct Case Relation Matching 115

3. Indirect Case Relation Matching (Phase 1) 117

4. Indirect Case Relation Matching (Phase 2) 117

5. Match Results 120

E. SUMMARY 120

VI. IMPLEMENTATION & PERFORMANCE 121

A. INTRODUCTION 121

B. MARIESEARCH 122

C. MARIEKEYS 123

D. MARIENLP 123

1. Natural Language Processing 123

a. Lexicon 123

b. Type Hierarchy 124

c. Parser 125

2. The Semantic Database 125

a. Keyword Index Files 126

b. Logical Form Files 126

3. Coarse-Grain Match 126

a. Setting up the Queues 126

b. Scheduling the Fine-Grain Match 127

c. Data Store 128

E. MARIEFINE 128
F. CAPTION ALTERATIONS 129
1. Case Sensitivity 129
2. Appositives 130
3. Commas 130
4. Missing Prepositions 131
5. Missing Nouns 132
6. Noun Ordering 133
7. Registration Data within Caption 133
8. Explicit versus Implicit use of "View" 134
G. PERFORMANCE MEASURES 135
1. Recall 135
2. Precision 136
3. Ease of Use 136
4. Response Time 137
5. Maintenance 139
H. SUMMARY 140

VII. CONCLUSION 142
A. INTRODUCTION 142
B. STRENGTHS AND WEAKNESSES 142
C. POSSIBLE EXTENSIONS 143
1. Unknown Words 143
2. NL Processing 143
3. User Models 144
4. Genetic Algorithms 145
D. SUMMARY 145

REFERENCES 146
APPENDIX A - ORIGINAL CAPTIONS 152
APPENDIX B - MODIFIED CAPTIONS 166
APPENDIX C - LEXICON 178
APPENDIX D - SYNTACTIC GRAMMAR 212
APPENDIX E - TYPE HIERARCHY 228
APPENDIX F - LOGICAL FORMS 250
APPENDIX G - EMPIRICAL STUDIES 329
INITIAL DISTRIBUTION LIST 356

LIST OF TABLES

TABLE 3.1. REGISTRATION DATA FOR CAPTIONS 262865-73. 16

TABLE 4.1. LOGICAL FORM RECORD FORMATS 75

TABLE 5.1. KEYPHRASE RECORD FORMATS 105

TABLE 5.2. KEYPHRASE RECORDS FOR CAPTIONS 262865-73 105

TABLE 6.1. MODULE SIZES. 121

TABLE 6.2. REGISTRATION DATA FOR CAPTIONS 218178-89. 134

TABLE 6.3. PARSE + MATCH RESPONSE TIME MEASURES 140

TABLE G.1. NL PROCESSING TIMES. 330

LIST OF FIGURES

Figure 3.1. MARIE Database Update Processing 17
Figure 3.2. MARIE Query Processing. 18
Figure 4.1. Type Hierarchy Design I - Certain concepts being treated as Class Instances 71
Figure 4.2. Type Hierarchy Design II - All concepts treated as classes. 72
Figure 5.1. Type Hierarchy & Graphs for Eg. 5.7. 113
Figure 5.2. Matching "view" in Query with Caption 5824 114
Figure 5.3. Matching using Correlation Link from Eg. 5.10. 118
Figure 5.4. Matching using Correlation Link from Eg. 5.11. 119
Figure 6.1. MARIE Client-Server Communication. 121
Figure 6.2. MarieSearch Window Environment 122
Figure 6.3. Semantic Database File Structure 125
Figure 6.4. Fine-Grain Matching (Time vs. Processors). 137
Figure 6.5. Fine-Grain Matching (Speedup vs. Processors). 138
Figure 6.6. Fine-Grain Matching (Speedup vs. Processors). 138
Figure 6.7. Log-Log Chart for Photo Lab queries (Query Length vs. Total Time). 141
Figure 6.8. Log-Log Chart for Photo Lab queries (Results Length vs. Total Time). 141

ACKNOWLEDGMENTS

I am grateful to Prof. Vincent Lum, Prof. Mantak Shing, Prof. Daniel Boger, and Prof. James Eagle at the Naval Postgraduate School (NPS) for their assistance these past several years in helping me prepare for and complete this research.

I am also grateful to Mr. William Ball and Mr. Bruce Bonbright for their support and encouragement while I was attending the NPS and during the time this research was performed at the Naval Air Warfare Center Weapons Division (NAWCWPNS) China Lake. I would also like to express my gratitude to the NAWCWPNS China Lake Long Term Training Committee for giving me the opportunity to attend the NPS to pursue the advanced training necessary to accomplish this research. I would like to say thank you to Lynn Cowley, the NAWCWPNS China Lake Photo Lab retrieval specialist, for her invaluable assistance during the last 19 months.

I would like to express my sincere thanks to Prof. Neil Rowe for all his support and encouragement prior to and during this research. His eagerness to experiment and debug the system, our numerous phone conversations, and constant Electronic Mail messages from the NPS helped keep me focused and motivated during the development at China Lake.

I. INTRODUCTION

A. INTELLIGENT IR AND MULTIMEDIA DB

Traditional information retrieval (IR) systems and most of the retrieval systems in use today rely largely on keyword, keyphrase, statistical methods, or some hybrid of these techniques. Keyword approaches use the existence or absence of certain words in a text to form the basis of relevancy to a user query; special words may also be used to separately index the subject area of a text. In this approach, the user is often forced to remember the valid words (i.e., keywords), how these keywords correlate with the concepts that he/she wishes to find, and how the keywords may be combined to formulate queries. Keyphrase approaches use predefined records to index the subject area of a text as well. Keyphrases are an extension to keywords that associate specific attributes with each keyword forming a record structure. For example, a keyphrase for keyword "aircraft" may have associated with it the type of aircraft, some alphanumeric designator, location of the aircraft at some point in time, and other pertinent information. Statistical methods are used in rating one text over another based on the search parameters in a user query, and using these rates as a measure of relevancy in determining what should be retrieved first, second, etc., or not at all. Such methods might discriminate one keyword over another through word frequency distributions in a text, word frequencies in user query requests, user or system predefined preferences for certain keywords over others, etc. In addition to these indexing approaches, the user must become familiar with the particular nuances of the IR system user interface. The NSF STIS User's Guide [NSF 91] contains a special tutorial section just on the TOPIC retrieval system.

Intelligent IR, as we perceive it, is the process of applying artificial intelligence (AI) techniques to information retrieval. Brooks (1987) defines an intelligent IR system as a "computer system that carries out intelligent retrieval" and that intelligent retrieval involves "... the use by a computer system of stored *knowledge* of its "world" (documents, users, topics, etc.) and of information about the *user* and his/her *problem* to *infer* which documents would enable that particular user to resolve or better manage his/her problem." Brooks provides a definition for intelligent retrieval as well. We have observed that recent approaches have used natural language (NL) understanding methods to improve the retrieval effectiveness -- the relevance of the answers to the query request. AI technology that can be incorporated into retrieval systems includes NL processing and understanding methods, rule-based (or expert system) reasoning, knowledge representation schemes such as blackboards or semantic networks, and others. Croft (1987) in his introductory article describes some of the basic approaches that have been taken in intelligent IR research, various difficult and open issues in the field, and the need for evaluation and testing of IR systems.

The evaluation and testing of the intelligent IR system must take into account the cost effectiveness of using AI techniques. If elaborate knowledge representation structures are incorporated, then we need to weigh any benefits that may be obtained against the time and cost it will take to encode these structures, either by retraining existing retrieval analysts, or hiring specialized knowledge engineers whose sole responsibility is building and maintaining the representation structures. Likewise, such techniques must show a cost saving in the amount of time expended by an ordinary user of the system. We conjecture that there will be an initial startup cost incurred by a new user of any new system. However, the interface should not require extensive knowledge about retrieval techniques, such as Boolean operators, probability estimation, relevance selectors, etc. It should also be dynamic in the sense to be able to adapt itself to the expertise of the user as usage increases.

We have applied basic AI techniques, specifically in the area of NL understanding, to implementing a cost effective solution for the intelligent IR of multimedia data. The focus of our research is to build an indexing and retrieval mechanism for multimedia databases (DBs). These consist of data elements that can take the form of either graphics, image, sound, text, or video. To identify a multimedia datum, we usually need registration information or fixed-field information to catalog and describe the datum. Such registration information can be stored in a relational database management system (RDBMS) and accessed using Structure Query Language (SQL) commands. In addition to registration data, we also often need captions to provide free-form information about the datum, to either describe an event occurring in it or unique characteristics and features. Captions then provide an alternative indexing mechanism for cataloguing the datum. A user should be able to find a datum using only NL if we can make the appropriate mapping to SQL.

Our research in intelligent IR then involves the development of a natural language processing (NLP) program and matching algorithms to support multimedia information retrieval. This report describes the current implementation status of our intelligent IR system to accomplish this objective. We have labeled the current system *MARIE* - *E*pistemological *I*nformation *R*etrieval *A*pplied to *M*ultimedia. We have chosen to develop this system for potential use at the Naval Air Warfare Center Weapons Division, China Lake, California for the indexing and retrieval of photographic images. The Center's Photo Lab maintains a database of over 100,000 photographs of project and historical data from the last 50 years. Both captions and *supercaptions* (caption about a set of captions) are associated with the data. We can characterize the linguistic structures of the captions as a *sublanguage* [Sager 86]. The current search and retrieval strategy uses manually created keyphrase records. We have taken a subset of this database to form the testbed for MARIE. Queries were also solicited from the database administrator to evaluate the effectiveness of our approach.

B. PROBLEM DEFINITION AND BASIC ASSUMPTIONS

Having briefly described the thrust of this research, we now define the problem. The research problem is to allow a user to run NL queries for retrieving multimedia data objects that have been catalogued using either fixed-field records or NL caption descriptions. There are four subproblems that must be solved. First, the NL captions and queries must be parsed into a logical form to arrive at a canonical semantic representation for matching. The canonical form reduces similar concepts to a single concept that captures their common meaning. This approach allows the matching to be based on common meanings rather than individual lexical items. Second, a coarse-grain match (keyword search) must be devised to confine the matching to those objects that are the most promising to avoid spending processing time on totally irrelevant objects. The rationale for confining the search to the most promising objects is not so much a requirement as it is a necessity; the size of the data can easily outgrow the size of any parallel processing system we could assign to the task. Further, there is no guarantee that a text will contain the exact keyword, or if it does, it may be the wrong word sense (definition) that is being used. For example, a user query may contain the keyword "plane" to denote an airplane where one text to be matched contains only the term "aircraft" while another contains the phrase "plane geometry." In addition, when a general term is used in a query, we not only need to find captions containing the general term, but also those captions containing specializations of the term; for example, a caption that contained "F/A-18" would match a query that contained "aircraft" since a F/A-18 is a kind of aircraft. Hence, the coarse-grain match must actually use canonical concepts and class/subclass concepts as the keywords.

In order to solve the second subproblem, the third subproblem arises. Index files must be devised for the caption logical form records, for otherwise we default to an examination of all of the logical form records each time a query is presented. In addition to examining the index files, the coarse-grain match must also search the registration data in the RDBMS using SQL "Select" statements if certain attributes can be inferred from the query. The results of the coarse-grain search against both data repositories can then be viewed as producing a list of the most promising multimedia data to investigate. If the matching stopped at this point, we would basically have a keyword search based on canonical concepts with the added advantage of searching predefined attributes in a relational table for the concepts as well. We have not examined how the concepts relate to each other. For example, if a query stated "missiles on an aircraft," the coarse-grain match would find all the captions involving "missile" and "aircraft," without considering the relationship "on."

The fourth subproblem is then to devise a more intense fine-grain match process to match the query logical form records against the most promising caption logical form records. For this problem, we need to investigate how the words in the query interact with one another to see if we can find a correspondence in the captions. This fine-grain match can be a slow process as we must compare the relationships and

concepts in both the query and each of the captions with respect to possible correlations, class/subclass relationships, and possible inferences that may be implied. Without a coarse-grain match to filter out the totally irrelevant captions, the fine-grain match would require an inordinate amount of time to complete all of the matching.

Some of the subproblems themselves are not trivial. As we mentioned earlier, despite all the work that has been done in NL processing and understanding, there still do not exist systems, present system included, that can truly handle any NL sentence that is thrown at them. In Rowe and Guglielmo (1991), we made the statement that NL captions are more formal than everyday English and hence should not be as difficult to parse and interpret. This statement is incorrect about some details. For example, in everyday English there is extensive use of determiners to help distinguish the word following them as nouns as opposed to verbs or other parts of speech. Parse rules to handle these structures when used in higher level parse rules may provide just enough of a preference for selecting one parse structure over another. In captions, the use of determiners is very minimal. Thus, more rigorous methods are needed to distinguish words that can take on multiple parts of speech.

Determining which attributes in a database relation pertain to a NL construct could prove equally as difficult, especially if we have multiple relations involved. For example, if we had the phrase "on Aug 12, 1983" in a query, then we could hypothesize that a date field in a relation is being referenced. This can easily be handled if there is only one date field; however, if there are multiple date fields that may also exist in multiple relations, the problem is nontrivial. Lastly, if we consider both the query logical form and caption logical forms as being labeled directed graphs, then we have a subgraph isomorphism problem. This problem is NP-Complete [Garey 79] and we must use some form of backtracking and maintain intermediate lists to know what matched what and what was the maximum score of all match combinations tried.

C. REPORT ORGANIZATION

In the following chapter, we present a survey of related work using NL processing techniques for information retrieval. In Chapter III, we present a conceptual overview of the system and describe the caption database that we have used together with modifications that were made to them. Chapter IV contains descriptions of techniques and problems encountered in parsing the captions as well as numerous examples. In Chapter V, we describe both the coarse- and fine-grain matching procedures that have been developed. Chapter VI describes the implementation and performance of the system with respect to the data structures and modules. Finally, in Chapter VII we describe various conclusions that can be made from the system and describe potential future research. The task of developing an adaptable user interface as mentioned earlier has not been incorporated into this initial work.

II. SURVEY OF PREVIOUS WORK

A. INTRODUCTION

Serious efforts to use NL and semantic primitives for intelligent IR are fairly recent; most research was started in the last fifteen years. In this chapter, we describe some of the related work on using some form of NL for IR. Traditionally, there are two types of text-based IR systems; those aimed at *document retrieval* and those aimed at *fact retrieval* [Findler 79]. In document retrieval, the user is concerned with retrieving a document that satisfies the criteria of the query. In fact retrieval, the focus is on understanding the text and one way is by question-answering. The system is supplied a text and attempts to understand the various concepts in it. The user then asks questions about what the system knows or can infer. Thus query results express (explain) the meaning of the information that was captured.

We can expand the notion of text in the previous paragraph to be any multimedia datum. The query results in our work are multimedia objects whose caption description matches the query description. This work is then similar to document retrieval; there is no attempt to explain the meaning of the caption description as is done in fact retrieval. We will shortly discuss some work involving NL text generation from picture analysis of static images and image sequences that can be used as the basis for retrieval. We have found that the majority of the systems investigated, while providing unique insights into their respective applications, are quite similar in certain respects. Retrieval effectiveness, however, varies as there is a large degree of world domain knowledge that is required to use NL techniques for intelligent IR and there are many different techniques that can be applied.

B. MULTIMEDIA DATA MODEL APPROACHES

A multimedia approach that has many ideas similar to our own is described by Damier et al. (1988). The authors described a multimedia data model based on object-oriented concepts including objects, a class lattice, state predicates, and a set of functions (as opposed to messages). The data manipulation language was an extension of SQL that supported path extensions (for referencing objects in the lattice) and generalized functions that can appear in either a SELECT or WHERE clause. Content search involved pattern matching operations consisting of Boolean operators, substring expressions, and an adjacency operator (ADJ) for finding two words in succession in a sentence (ADJ may require a transformation into Boolean operators). Signature files were used for representing a text data type. A signature file can be viewed as consisting of blocks where each file block has a signature block (a string of bits). A lexicon was used for differentiating common from non-common words. These non-common words were then hashed to the signature block. The signature blocks were used for content search. Hence, depending on the hashing

functioning, number of collisions, and transformation loss, irrelevant documents may be retrieved. A relational DBMS was used to hold the class structure, catalogs, and object descriptions. The actual multimedia data, signature files, and lexicon were stored outside the DBMS in regular UNIX files.

Bertino (1988) discussed the problem of query processing in a multimedia document server for the MULTOS project. As he pointed out, query processing must take into consideration the fact that the multimedia data can be stored both on magnetic and optical storage, requiring different access and storage models. For query purposes, a document was described by its *conceptual components* which are areas in the document used for specific purposes -- address in a letter, title, and author information are given as examples. Retrieval commands consisted of FIND and WHERE clauses where NL can be used for substring expression retrievals in one of two ways: either against predefined *components* (fields) or against general *text* fields in all documents. Text fields were indexed similarly to Damier's approach, i.e., using signature files.

A search based on *partial* and *exact* match for retrieving multimedia office documents was described by Croft and Krovetz (1988). Matching against text fields resulted in partial match scores with values from 0 to 1; matching against predefined field attributes were regarded as exact matches with values either 0 or 1. Uncertain query specification was supported which allowed a user to indicate which attribute values were PREFERRED or ACCEPTABLE as well as which attributes were of HIGH, MEDIUM, or LOW importance. The user queried the database by specifying attribute values in predefined templates where the templates corresponded to various document types.

Christodoulakis and Graham (1988) described a system for browsing through "time driven" multimedia documents such as sound, video, and animation data. Such data can be organized into a directed acyclic graph for presentation purposes. "Hidden text" was used to describe each node to allow random access to any node in the graph. This text could correspond to keywords, keyword sequences, replicas of a text portion, or even a caption. However, there was no NL processing component to locate a hidden text segment -- the user was forced into remembering keywords. To remedy this problem, icons were associated with each node to represent the node's content.

At a workshop on Intelligent Multimedia Interfaces, Davis (1991) pointed out the need for semantic representation of multimedia content because of the numerous inadequacies of keywords. NL descriptions (or captions) would serve as the labeling component for the data. Davis also pointed out the need to describe the temporal properties of events within the data, such as when a particular scene segment appears in a video, how long the segment is displayed, camera angles, etc. A set of descriptive layers was proposed to describe a segment from which the content of the descriptors can be considered small captions as well.

Baudin et al. (1991) discussed the use of design records to describe the content and form of design information for mechanical devices. The user was allowed to retrieve any of several multimedia types

depending on the descriptors entered. An example mentioned was the assembly of a device where a picture would be more valuable than a text description. A taxonomy of predefined classes was used and retrieval was based on simple inheritance and transitivity rules. A description of these rules can be found in Rowe (1988).

C. ENHANCED KEYWORD APPROACHES

We use the term "Enhanced Keyword" to refer to those IR systems that have applied AI techniques to keywords and keyphrases. An expert system, GRANT, for finding research funding sources was developed by Cohen and Kjeldsen (1987). The system demonstrated the feasibility of constrained spreading activation for information retrieval. A semantic network was used, where nodes represented the funding agencies and the research topics that were supported. The nodes can be viewed as keywords. Slots were used to provide access to research specifics as well as relationships to other research. Research goals could be classified by one or more of ten classes, similar in nature to the ACTs defined by Schank (1975). Spreading activation in the search was controlled by fixing on a link distance and placing a bound on node fan-out to limit the amount of expansion in the semantic network. Path endorsements were used to prune and order search paths. If an exact funding match could not be found, GRANT attempted to find related funding sources (partial match). Performance of 29% precision and 67% recall was reported.

The use of a hierarchical structure where the user can enter keyword phrases to move to any particular node was described by Smith et al. (1989). EP-X was a knowledge-based system for the retrieval of bibliographic information on environmental pollution. Information was organized into hierarchical concepts where each node in the hierarchy contained a list of semantic primitives serving as document identifiers. Frames were also attached to the nodes to indicate how concepts were used and to specify relationships among concepts. Concepts expounded upon were the use of tangled hierarchies based on alternative organizing perspectives by the user, the ability for a user to examine the hierarchy in a top-down fashion or move directly to the level of interest, and the ability to modify and learn the hierarchies when new information was introduced into the hierarchy. A slightly different approach involves traversing keyphrase hierarchies [Ragusa 90].

An approach based on handling vocabulary issues is "latent semantic indexing" (LSI) [Dumais 88]. As the authors point out, the basic semantic problem is that users want to retrieve the desired information based on meaning, yet the words they use may not accurately capture the meaning. In LSI, single-value decomposition was used. Terms (words) that appear in more than one text-object (caption, multi-paragraph units, documents, etc.) were mapped to a term by a text-object matrix. Analysis of the matrix was performed to arrive at a reduced table that approximated the underlying semantic concepts of the original matrix. The matrix can then be viewed as a 2-D space. User queries were mapped to this space as well and matches were defined by a conical region within proximity of the query. Performance statistics showed

some performance improvements over a keyword search. Similar type work was described by Wong et. al (1987) where a vector space representation was used for modeling documents and queries.

Anick (1991) described their approach to augment the STARS retrieval system at Digital Equipment Corporation. NL processing was used to augment the traditional Boolean searches. Users could enter queries in NL or in Boolean expressions. Some of the techniques used can be viewed as augmenting SQL based-searches. No semantic analysis of the NL query was performed. The system strictly looked for keyword terms to search on and from there formed Boolean query expressions. The English phrases entered or manipulated were error message descriptions or titles to various instructions. Synonyms were used explicitly instead of mapping to one canonical representation for all synonyms.

Driscoll (1991) described a system that can automatically create keyphrases using "template phrases." This system, JAKS, indexed military command data (e.g., lessons learned) for a retrieval system for the Joint Chiefs of Staff. The process involved scanning the documents for potential phrases, deleting ambiguous phrases, implying additional phrases, and selecting those phrases that match a template for keyphrase generation. The system was evaluated against a team of subject-matter experts for manual and automatic creation of keyphrases with respect to assignment and accuracy. Performance was mixed on new documents but improved when the experts went back and adjusted the rules for those documents. Driscoll also provided a history of additional automated keyword indexing approaches.

D. NL APPROACHES

The rationale and motivation for using NL captions for the handling of multimedia data was presented by Lum and Meyer-Wegener (1989, 1990). It was envisioned that captions could serve as the indexing information for the multimedia data. The design described in the reports called for the creation of predicates interconnected using object identifiers with associated inference rules. Dulle (1990) demonstrated the feasibility of this NL approach by developing a rudimentary parser for handling selected captions from photographs taken during World War II. Grammar rules and a preliminary list of predicates were defined for handling the captions. This prototype parser was useful in demonstrating how NL queries could be used in conjunction with SQL for specifying retrieval requests from a multimedia database. The parser implementation, however, turned out to be quite inefficient and extremely slow in processing the captions, and hence was not carried forth in this work. Further analysis and processing strategies for using captions were described in Rowe and Guglielmo (1991).

As we expressed earlier, there are many issues and techniques for using and applying NL for IR. Some of these issues include deciding how much syntactic and semantic information should be applied, should every single word be parsed and understood or should just the major concepts be explored, what is the best

knowledge representation, etc. These topics and others have been addressed by almost all NL researchers in developing an IR system; many of these issues are still open.

Findler (1979) proposed a system, IRUHS-1, using a hierarchical associative network for organizing knowledge for question-answering. He identified three levels in this hierarchy. The lowest level consisted of *nodes* of atomic pieces of information that could be provided by the user or computed, an example being a particular athletic event in the modern Olympic games. The next level consisted of *clusters* for representing similar information where the nodes of a cluster could represent facts concerning the athletic events of the 1976 Olympic Games. At the highest level of grouping, a *plane* allowed clusters to be grouped based on similarity. Nodes could belong to several clusters, and a cluster to several planes. Findler raised several open issues dealing with methods for forming planes and clusters; Boolean combinations being two such methods. NL queries were planned, but no implementation details were done at the time.

The organization of a conceptual memory for retrieving facts by intelligent IR systems was first explored by Kolodner (1983). CYRUS received as input news story summaries and stored biographical information about important people such as Cyrus Vance and Edmund Muskie for subsequent retrieval. A different system, FRUMP, was used to skim the news-wire, analyze new stories, and pass the story summaries on the subject area to CYRUS. The events were represented using Conceptual Dependency case frames and scripts [Schank 1975, 1977]. CYRUS used a high-level knowledge structure, E-MOPs (Extended Memory Organization Points), to organize (categorize) the events. Indexing of events was done by specifying the E-MOP and a feature-value for some unique feature within the category. Since unique features could not always be guaranteed, E-MOP subcategories were introduced. Queries could be entered in NL and a semantic representation would be produced. However, the representation might not include the specific E-MOP to be searched or the features specified in the query might not be applicable for a given E-MOP. To handle this problem, two exception procedures were devised. If a problem still arose, then alternate search keys needed to be generated. CYRUS was an attempt to demonstrate the feasibility of integrating NL knowledge structures in IR; performance was not an issue at the time. The problem was understanding the knowledge mechanisms needed for indexing and retrieval.

Rau (1987) developed a similar retrieval system termed SCISOR. SCISOR selected and analyzed newspaper stories involving corporate mergers and takeovers. Problems in dealing with multiple knowledge representations as well as a lack of inheritance in CYRUS were overcome through the use of the KODIAK knowledge representations [Wilensky 86]. This representation supports episodic, abstract, and semantic memory. The parser processed each word of the sentence to derive a semantic representation; alternative syntactic structures were derived by the semantic interpreter and the most promising was selected. Retrieval was based on a two-stage process involving a coarse- and fine-search. The coarse-search involved a constrained spreading activation such that when a certain subset of instances was created in the memory, the

entire episode was brought into memory for the fine-search. We can in turn view these episodes as caption representations. The fine-search involved checking the NL query graph against the episode graph-matching linguistic case structures and values; partial matching results were also available. As an understanding system, SCISOR was capable of answering questions about information explicitly stored in the knowledge base, but not implicit information. Generalization of events was proposed to facilitate learning. Jacobs & Rau [90] listed performance in a constrained environment (constrained in the sense of corporate takeovers and mergers) at 80-90% combined recall and precision. Meaningless results would be produced if the environment was not so constrained to that domain.

Vickery and Brooks (1987) described an expert system, PLEXUS, to serve as a referral system on gardening topics for use in public libraries. PLEXUS allowed a user to enter anything from keywords up to a NL sentence in order to specify the problem statement. From there, it filtered out those words that may not contribute semantically to the problem domain and created term (keyword) lists for searching the lexicon (a syntactic parser is not used); a type hierarchy of terms was used in the process. The system then checked the words to determine if additional information may be needed in order to create a semantic representation of the query. Eleven (11) semantic categories were used for a case grammar representation. The representation consisted of a problem statement model composed of frames, one frame for each term. The objective was to fill those slots necessary to initiate a search. Rules were used to determine completeness and the user was prompted for additional information if a necessary slot needed to be entered. The user also had the ability to examine the type hierarchy in attempting to determine what to enter. The search expression consisted of the frame slot values combined using Boolean operators that, together with terms, can be added, modified, or deleted depending on the search results. Once the search was completed, the system explained the results to the user.

Katz (1988) described the START system whose knowledge base can be used for question-answering or as input to another system. The START system took a sentence and decomposed it into smaller units. Ternary expressions (*T-expressions*) were used to capture the <subject, relation, object> features of a sentence. Transformational rules and embedding of one T-expression inside another were used to handle complex sentences. Matching involved translating NL queries into T-expressions and matching them against stored T-expressions. We can view these T-expressions as being two-argument predicates where the predicate name expresses the relation. START returned all expressions matching the query expression.

Translation of NL queries and documents into first order predicate calculus expressions with Prolog matching and unification was investigated by Sembok and van Rijsbergen (1990). The parsing approach was based on a Montague grammar. An advantage is that no knowledge of the domain is needed in order to derive a semantic representation. The grammar developed handled only restricted noun phrases. Since the motivation was not to produce a question-answering system, the focus of the work was to produce good

content indicators or indexes for the queries and documents, and not worry about exact meaning. When encountering a noun phrase, each noun and adjective was treated as a separate predicate. Hence, both the queries and documents were translated into logical forms and then matched; predicate names in the logical form corresponded to stemmed English words. We should note that redundancy is introduced because there is no simplification that can be applied to the noun phrase because no domain knowledge is known. When dealing with descriptive captions, we will see certain advantages in simplifying this type of structure because we have domain knowledge available. Indices were created for the predicate names and each term was given a statistical weight based on its frequency in a document; nominal compound weights were computed as average weights of their constituents. Tests were conducted on *Communications of the ACM* articles with 48 standard queries chosen from a set of 64. Results presented indicated an 8.2% increase in precision over previous benchmarks.

Dick and Hirst (1991) described the retrieval problem of case law reports. Sowa's conceptual graph structures (1984) were used to capture the semantic representation. As the authors pointed out, law reports in general are very dense writings and extracting the technical meanings can be a laborious task. Understanding many of the nominal compounds and noun phrase interactions is also a laborious task. The authors discussed many of the representation problems encountered including replacing Sowa's linguistic cases because of certain inadequacies -- dual roles being one example. Matching was accomplished through frame matching involving marker passing and controlled spreading activation.

Text skimming is an approach that has been taken by a number of researchers in the last few years. The idea being to identify beforehand various words or patterns of words that are of interest [Hilster 1991]. The skimmer would then attempt to find a region within a text that appears to contain the interest points, and then see what types of information may be in the region in order to enter information into various pre-defined templates. The rationale for using skimming approaches is when there is a large amount of text information and the user is trying to find if there are potential items of interest in the text. Thus, in-depth conceptual processing of every word in the text is not needed. Slot values for the templates show this as well, as these values are not analyzed down to a specific word, but rather remain at a phrase level. When using captions for our text basis, what we are after is an understanding of what the objects are in the multimedia datum and how they are interacting with one another. Being able to identify relationships and categories of objects is extremely important in our application. Further, we assume that there is not much text in the caption making in-depth analysis feasible. Hilster's system processed messages about terrorist activities and filled in predefined templates to describe the events. The next step was to process user queries against these templates. This step, however, was not discussed at the time of the writing. Finin (1991) also discussed a similar type approach.

Mauldin's work (1991) was also a text skimming approach and parallels our own work in many respects. The problem domain dealt with full text retrieval and the introductory chapter introduced the "keyword barrier" problem. The semantic representation chosen was CD and scripts, similar to the approach of Kolodner. Potential problems in both systems are the lack of suitable scripts for understanding. If the system is scanning all texts, then scripts must be available for every possible situation that can be expected to be related to the original query. The lexical knowledge base had an interface to Webster's On-line Dictionary and created frames for each word as needed; multiple word senses can be handled and a type hierarchy was also incorporated (and used for matching). A user did not enter NL queries, but CD or partially instantiated scripts instead. Mauldin suggested an inverted file method to improve performance during matching. We have followed a similar strategy with some modifications. The method he proposed introduced redundant information and increased storage. Matching results were binary; either the query matched a script or it didn't -- there was no partial matching. The use of genetic algorithms was described to improve performance by modifying the scripts (learning).

E. PARALLEL MATCH APPROACHES

Stanfill and Kahle (1986) described free-text searching using a massively parallel computer, the Connection Machine. *Surrogate coding* (a hashing technique used for spell checking) was used to represent a document in a processor memory. However, several documents could hash to the same combination and result in false hits. Furthermore, if a document contained more than the allocated number of words, several tables needed to be used. Hence, from one to three documents could be stored in a processor at one time depending on the number of tables used. There were several ways queries could be performed. In the first method, queries consisted of terms of interest (not necessarily keywords) and combined into a Boolean expression. The second method involved using *simple queries* consisting of a list of terms where each term was assigned a point value; search then involved coming up with a total score for each document and performing normalization of the scores. The last method involved *relevance feedback* where the simple query method was used to initiate the search. Once the initial resulting documents were obtained, the documents were marked good and bad with respect to relevance of the query. Then the good documents were scanned to arrive at a second simple query statement. This query could then be used as the input to another search and the cycle repeated until the user was satisfied with the search results. High recall and precision values were reported for the relevance feedback approach.

Gallant (1991) emphasized the importance of a representation that is both time and space efficient with respect to the size of the data. An interesting observation of this work was the potential mapping to neural network architectures. He advocated the use of context vectors to represent words, documents, and queries. Context vectors were used to describe an information source against a list of predefined features (we can

consider these features to be words). Thus, both a document and a query were described by their context vectors and matching involved accentuating the similarities between the two vectors. There was no NL component provided - queries consisted of any number of terms or documents retrieved. Feature list(s) were created for a particular domain and each new word, document, or query needed to be defined manually. This should not be construed as a drawback; even in our system, a lexicon and type hierarchy must be constructed manually for a particular domain.

F. IMAGE ANALYSIS APPROACHES

Image researchers are investigating the best methods to describe images. Three problem areas from this research that have bearing on our work are the description of static images, dynamic images, and image sequences. The static and dynamic image problems are discussed by DiManzo et al. (1986). The static problem involves defining a set of spatial relationship primitives for the representation of objects in a picture; for example, "by the table," "next to the right," etc., where the focus is mostly on discovering how the preposition qualifies the noun phrase that follows and what relationships can inherit. The dynamic image problem deals with describing events and touches upon the NL problem of story understanding. Captions fall into this category. The image sequence problem [Nagel 88] concerns itself with spatial relationships during a time sequence relating to video data.

Chang (1988) described a method whereby pictures and queries can be represented by a *2-D string* consisting of orthogonal and diagonal spatial relations. The process involved identifying and labeling the objects in the image, then defining spatial relationships between "point of view objects." Queries then involved spatial relation type questions; for example, find all images where an aircraft is flying over a mountain or find an image where a missile is near an aircraft. These queries involved 2-D string subsequence matching. Queries could be specified verbally or graphically by drawing icons on the screen representing each object of interest and showing the spatial relationships.

Roussopoulos et al (1988) provided for user querying through extended SQL as well as a graphical interface. Object-oriented DBMS's were also proposed for representing spatial relationships [Mohan 88]. However, these two approaches did not employ any NL semantic representation of an image description.

System-generated NL descriptions of an image sequence have been investigated by a number of researchers [Neumann 83], [Niemann 84], [Novak 86], [Schirra 87], [Andre 88], and [Nagel 88]. Such systems provide the user the ability to ask questions with respect to static and dynamic relationships and kinematics. In addition, by controlling the NL generation, there is more control on the semantic representation. Neumann introduced the use of event models for generic scene descriptions. These models consisted of verbs that identified a type of change and those predicates that were used to describe the event. Neumann (1984) then discussed mapping the deep case structure of NL queries into predicates that can be

matched against stored events. These predicates consisted of spatial relationships and some linguistic case structures. The results are similar to some of the research involving story understanding -- in this case though, image sequence understanding is being examined.

G. OTHER APPROACHES

Hypertext systems with the ability to adapt a specific user's model to the context (view) he wishes to see has been explored by Boy and Paris (1991) with the domain being Space Shuttle operations and Space Station Freedom. Clifton and Garcia-Molina (1990) discussed an approach for indexing in hypertext databases where the user can specify queries consisting of a start point, the type of link to follow, and a test in order to determine if the object at the end of the link satisfies a certain condition; this type of indexing amounts to following a directed graph. The uses of a *text graph* to represent the thematic topics of a full-text document was described by Hahn and Reimer (1988). The text graph is organized hierarchically with general information at the root and specific information at the leaves. Gordon (1988) demonstrated the usefulness of genetic algorithms for document re description where the keyword representation was modified to reflect the manner in which queries were stated by a group of users (relevance feedback). Zellweger (1988) described a scripting mechanism for organizing and retrieving multimedia documents for playback presentation and animation.

Transportable NL interfaces that can be used as front-ends for querying DBMS's have been investigated by Hendirx et al (1978), Kaplan (1884), and Grosz et al (1987).

H. SUMMARY

We have found that there has been much work on various aspects of intelligent IR. Our NLP of captions is fundamentally different from previous NLP approaches. Captions contain very rich noun phrase structures containing class/subclass concepts that can be used to simplify the noun phrase, correlations, actor and ownership information, measures, and basic adjectival modifiers among others. From the noun phrases, we can infer events stemming from verbal type nouns and nouns interconnected by certain prepositions. Unlike previous verb-driven approaches, focusing on the major verbs for understanding is not appropriate as very little verbs are found in a caption. Captions may also contain spatial relationships about the objects found in images, both explicitly stated and implicitly understood, as well as describe a short story similar to what we saw earlier. In processing the captions, we have had to perform detailed examination for each of the words in a sentence or phrase to see how they interact with each other, something previous approaches have appeared to do minimally. We have found that certain rules can be generalized to apply to any domain, while others are constrained to the specific domain and situations of NAWCWPNS, China Lake. We have not discovered previous attempts that use captions or caption-like

descriptions for intelligent IR. We believe our approach provides a common method for indexing and retrieving not only images, but all forms of multimedia data.

III. SYSTEM APPROACH

This chapter describes a brief overview of the processing and data files for MARIE as well as the caption data that was used to populate and test the system.

A. INTRODUCTION

The Naval Air Warfare Center Weapons Division (NAWCWPNS) China Lake Photo Lab maintains a database of over 100,000 photographs of project and historical data from the last 50 years. An Ingres database is currently used to catalog and support retrieval of registration data pertaining to the photographs. Two relations are used: *visual* and *keyphrases*. The *visual* relation maintains the registration data for a photograph or a set of photographs. Table 3.1 shows a sample record from the *visual* relation. The *id* indicates that this registration record applies to a set of photographs, specifically 262865 through 262873. Moreover, the caption is written in such a way to make it applicable to all nine photographs. The *keyphrases* relation will be discussed in Chapter V. Appendices A and B contain the caption descriptions.

TABLE 3.1. REGISTRATION DATA FOR CAPTIONS 262865-73.

ATTRIBUTE	VALUE
Designator	LHL
Id	262865-73
Quantity	9
Date-Orig	09-apr-1990
Retention	H
Medium-Info	645 VERI
Photographer	D. CORNELIUS
Customer	CUSTOMER SERV
Code	34501
Location	ARMITAGE
Date-Loaded	26-jul-1990
Caption	SIDEWINDER AIM-9R MISSILE ON A STAND AND VIEWS MOUNTED ON AN F/A-18C BU# 163284 AIRCRAFT, NOSE 110. LHL 262867-68 WERE RELEASED BY L. KING ON 07-24-90.
Class	U
Cross-Ref	

We used these records as a starting point in the development of our system. A view of the MARIE database update processing is shown in Figure 3.1. The administrator interacts with the system using a simple line-driven user interface. The interface sends a natural language (NL) caption to the NL processing (NLP) subsystem for processing. Using the domain lexicon and type hierarchy, the NLP subsystem parse the query into logical form (LF) records [Allen 87] and then send the caption LF records to the "make keys" routines. These routines determine the noun and verb concepts and update the noun/verb indices that are used

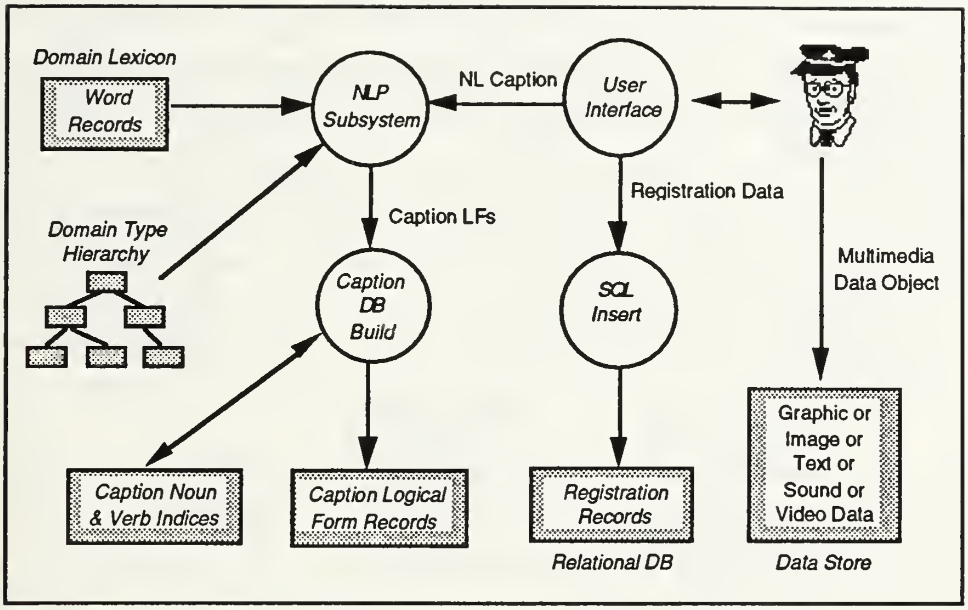

Figure 3.1. MARIE Database Update Processing.

by the coarse-grain match. The caption LF records are also stored for the fine-grain match. Registration type data is added into a relational database using the SQL "Insert" command. Each multimedia data object is stored in its own file.

Figure 3.2 shows the query processing view. A graphical user interface (GUI) was built instead of a line-driven one to make the system more user-friendly. Similar to the database update processing, the GUI sends a NL query to the NLP subsystem for processing. The query LF records produced is now sent to the coarse-grain match that selects the nouns and verbs from the records together with class/subclass information from the type hierarchy to determine the keywords to examine. The registration records are queried using Structured Query Language (SQL) commands while the noun/verb index files are merged in to one list and intersected with the SQL results. The most promising caption identifiers are then sent to the fine-grain match for detailed examination along with the query LF records. The fine-grain match uses the type hierarchy and the logical form records in its processing to determine a match score for each of the captions. The caption identifiers and match scores can then be sent to the GUI for presentation to the user. The user is able to select the multimedia and registration data for any of the caption identifiers displayed.

We now discuss the data and processes that comprise the major system components in more detail.

B. DATA FILES

We view the data files for MARIE as being grouped into three components. The first component is a relational database to store and retrieve registration information. The second component is a collection of

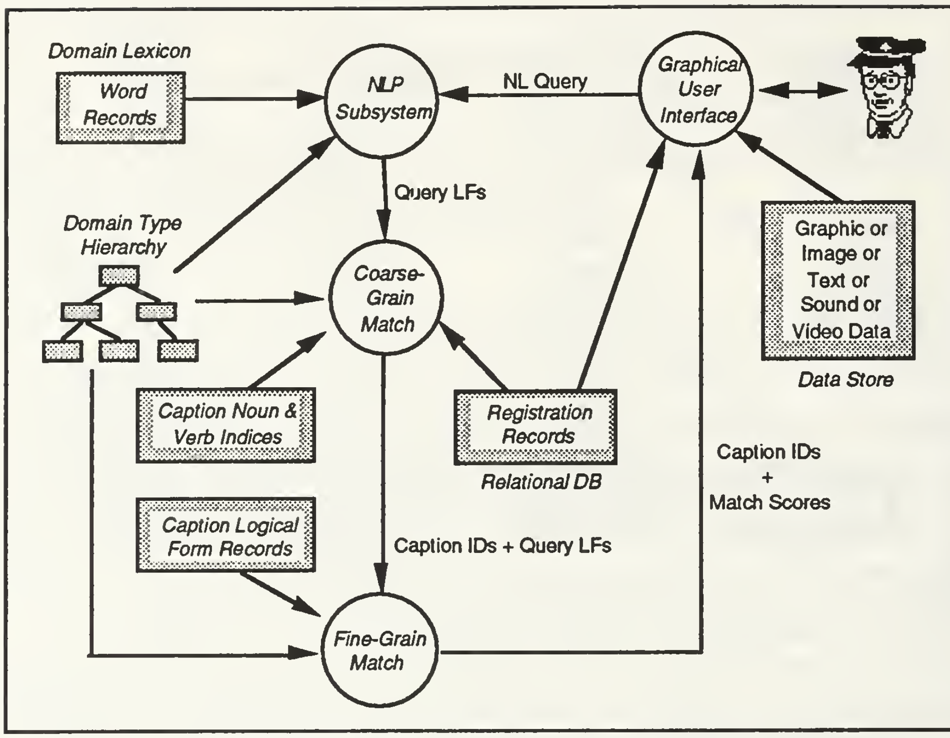

Figure 3.2. MARIE Query Processing.

files used for parsing the NL captions and storing the resulting LF records and indices, which we term the *semantic* database. The third component is a data store that contains the multimedia data itself.

1. Relational Database

The relational database is used for storing the registration data for a multimedia data object (e.g., location for an event, date, time, image size, animation length, etc.) as well as textual information (e.g., personnel records). The database administrator would have to decide what attributes are required to handle the registration data as individual items and what should be included in the NL caption. Once the relational structure is designed, the registration data can be entered and queried using SQL commands. We envisioned that any commercial relational database management system (RDBMS) could be used to fill this role. Each registration data record would contain a unique identifier that maps to either one or to a set of multimedia data objects. For example, in cataloguing a photograph sequence, the indexer may generate one registration record for the set instead of describing the characteristics of each photograph in separate captions. One possible naming convention for the identifiers was described in Holtkamp and Lum (1990). Table 3.1 presents an acceptable example of a registration record for our system.

2. Semantic Database

The semantic database holds all the data files necessary for processing and handling the NL access of the multimedia data. In order to understand any query or caption, a lexicon and type hierarchy must first

be defined. The lexicon defines all the words and senses to be used in a particular domain and includes for each word its part-of-speech and morphological features, canonical representation, and possible case information when defining verbs. The type hierarchy identifies how the concepts are hierarchically ordered (e.g., F/A-18A is *a kind of* aircraft), as well as their correlations (e.g., aircraft *has a* wing). The original plan [Lum 89] was to use simple predicate calculus expressions in a manner similar to Sembok and van Rijsbergen (1990). An example of a predicate calculus expression is shown in Eg. 3.1. In such a scheme,

Example 3.1. Predicate Calculus Representation for "the red aircraft banked left."

```
aircraft(X) ∧ red(X) ∧ left(Y) ∧ bank(X, Y)
```

predicate names for the caption and query terms could be classes in the type hierarchy as well as any form of modifiers. Some of these names, however, conflicted with additional predicate names that we believed necessary to represent roles. Almost all previous NL understanding methods we investigated used some form of case grammar representations that impose a structure on predicate calculus forms. By adopting such a methodology, we were able to visualize the matching task easier. Hence, we simply opted for a case grammar representation where the predicate names designate the different cases. A case grammar representation is shown in Eg. 3.2.

Example 3.2. Case Grammar Representation for "the red aircraft banked left."

```
inst(X, aircraft) ∧ attribute(X, red) ∧ event(Y, bank) ∧ agent(Y, X)
direction(Y, left)
```

Index files in keyword systems would correspond to inverted files. To generate indices for our system, the NLP module must first parse the captions and arrive at a set of LF records that represent the meaning of the caption. From these records, we can create index records containing the nouns and verbs found in the caption. These records provide our coarse-search index, similar to the concept described by Rau [87] and Mauldin [91]. Eg. 3.3 shows possible index records for the caption in Table 3.1. Each keyword could have its own file where the file consists of those caption identifiers where the keyword appeared. However, by analyzing noun phrase structures, we can reduce the number of keyword files and entries required; this is discussed further in Chapter V. In addition to the indices, we also store the caption's LF records for further analysis by the fine-grain match. The records in Eg. 3.2 could represent the LF records

Example 3.3. Index records for caption of Table 3.1.

```
(SIDEWINDER, 262865-73)    (AIM-9R, 262865-73)       (MISSILE, 262865-73)
(STAND, 262865-73)         (VIEW, 262865-73)         (MOUNT, 262865-73)
(F/A-18C, 262865-73)       (BU#163284, 262865-73)    (AIRCRAFT, 262865-73)
(NOSE-110, 262865-73)
```

for a particular caption. We have the option of storing both the indices and LF records in the RDBMS or separately in individual files.

3. Data Store

The data store (i.e., multimedia data store) is used to hold the actual graphic, image, sound, text, and video data. Because of the size and format of this data, it is more feasible to store each data element in its own separate file. We have several options on how the data can be categorized based on its type. Also, either the registration data must be used to identify the multimedia types or we must devise a standard file naming convention and/or directory structure for identification. The reason is that we have to know which program to invoke to display or play a multimedia datum. For example, if a multimedia datum was stored in the file "xyz," whether the contents of this file are an image or sound file can be determined by examining either the registration data or the file name if some mnemonic was used.

File formats for a multimedia type are another important issue. For images, there are a number of formats that can be used including PICT, TIFF, GIF, XBM, RAS, etc. Holtkamp and Lum suggested a class hierarchy based on file formats. We believe that this is not necessary. There are a number of conversion tools, both public domain and commercial, that will convert from one file format to another. The San Diego Supercomputer Center provides one such suite of tools. Hence, we can represent the data in one standard format and convert to appropriate file formats as needed. The choice of file format can be made based on the size of the files needed to represent them. For example, we have discovered that an image file in GIF format requires less storage than most file formats. In the case of image display, by standardizing on the X-Windows environment, we can display images to all UNIX workstations, Apple MACINTOSHES, PCs, and PC clones.

One other decision to be made involves choosing the storage medium type that, for the most part, is either optical or magnetic. With current technology, it is more practical to store the volatile information and indices on magnetic storage because of the faster access speed. The actual multimedia data can then be stored either to magnetic or optical depending on file size and the frequency with which it needs to be accessed.

C. PROCESSES

1. NLP Subsystem

The NLP subsystem handles the parsing of captions and queries. As we saw in the previous chapter, there are many different ways NL can be used. The approach we have taken attempts close to full NL understanding; we have attempted to classify and analyze all of the words encountered in both captions and queries. There are several reasons for this, mostly derived from analyzing images. First, the

photographic captions we have encountered have very rich noun phrases and a multitude of nominal compounds. Many captions consist of nothing more than noun phrases (i.e., no verbs). Hence, we need to understand what is happening in the noun phrase in a way similar to interactions expressed by verbs. Second, we need to understand the spatial relationships between the objects as described, for example, a "missile under wing" versus "missile away from aircraft." Third, we need to be able to infer additional relationships. In the first example above, the fact that a missile is under the wing can be taken to imply that the missile is also on the aircraft. Lastly, captions themselves do not tend to be very long, averaging 20.7 words per caption. Hence, processing times are not excessive.

A problem we sought to avoid was the requirement to build pre-defined knowledge structures to avoid the problems encountered by Mauldin (1991). The problem with scripts is that someone has to handcode the applicable ones for each domain in addition to setting up the lexicon and type hierarchy. In our environment, there is a shortage of trained personnel who have the skills to create such a knowledge structure. Hence, we need to keep the maintenance of the system simple. Thus, our NL program needed to be designed so that there are no predefined templates or scripts that need to be created. In essence, we have a neutral logical form representation that does not provide a complete description of the events in a caption and may not answer some query questions correctly.

The processing structure of the NLP can be defined simply as a blackbox. The inputs are a lexicon, type hierarchy, and caption or query. The output is a set of logical form records. There should be no difference between the way the caption and query logical form records are created. By using case frame (grammar) record formats and following a *slot-assertion* type format [Charniak 87], we then have a representation that appears similar to a graph. Since both the query and captions appear as graphs, matching involves determining a subgraph isomorphism.

2. Coarse-Grain Match

The coarse-grain match is an enhanced keyword lookup approach that forms the first phase of the matching process. Based on the nouns and verbs processed in the user query, we attempt to find all captions that contain the query nouns and verbs, or their subtypes as defined in the type hierarchy. In addition, we also perform some semantic analysis on the way the nouns appear in the query in order to determine if they might be contained within the registration data. For example, if a query statement contains location information, we have a mapping function that will examine the *location* field (if one exists) in the registration data using an SQL *select* statement. It will then examine the *location* records in the caption logical forms. The resulting caption identifiers are then combined ("union'd") with the caption identifiers from the other nouns and verbs to arrive at a keyword score. These scores are then thresholded against a coarse-grain threshold to see which captions are candidates for the next matching phase.

3. Fine-Grain Match

The fine-grain match forms the second matching phase and entails mapping the logical form for a stored parsed caption back into the type hierarchy and matching it against the query logical form also contained within the hierarchy. As we indicated earlier, both the query and caption logical forms can be viewed as graphs where the vertices are the noun and verb instances and the edges are the case relations. In addition, each node can be viewed as being a tree node in the type hierarchy, giving a somewhat three-dimensional view. The task is to determine if the caption logical form contains a subgraph of the query logical form. For example, if the query is "missile on an F/A-18" and the caption reads "Sidewinder mounted on a wing of an F/A-18A," then clearly the caption graph has a subgraph corresponding to the query graph. It is also possible that the caption graph has a subgraph corresponding to a subgraph of the query graph, which results in a partial match. As this example showed, instance matching (matching of vertices) is based on subtype matching (e.g., missile ⇔ Sidewinder). Matching of relationships (edges) entails using a predefined set of relationships and similarity matching.

4. Database Update Routines

Given the three categories of data files discussed in the last section, we need appropriate database update routines for each. Updating the domain lexicon and type hierarchy are both manual processes that require the database administrator to add new terms by editing the files. The caption database build function for a caption is conceptually similar to the coarse-grain match for the query. It extracts the nouns and verbs from the caption LF and updates the associated index files; it also saves each entire caption LF. Updating the relational database can be done using the SQL "Insert" command for a particular commercial DBMS. Creating the image files for the data store requires scanning in an image and storing the image in an appropriate file format.

5. User Interface Routines

Both user interfaces are relatively simple and straightforward. The database administrators interface is constructed to handle batch processing of caption records. The query interface allows a user to enter the NL query. It then presents a list of the captions that matched together with scores and allows the user to examine a datum and the accompanying registration data.

D. A SAMPLE RUN

The following program trace illustrates the NL and matching processing. The query "aircraft with an AIM-9R missile" is provided as input together with the multimedia category image. The end results are the maximum possible keyword score, maximum possible match score, and a list of caption identifiers and scores.

```
Query: aircraft with AIM-9R missile

Received:
captionSearch(image,[97,105,114,99,114,97,102,116,32,119,105,116,104,
32,65,73,77,45,57,82,32,109,105,115,115,105,108,101])
"aircraft"
"with"
"-9"
"missile"
End - processBody
Found paragraph 1, sentence 1

Creating parse trees from lexicalizations.

   Sample 1   Dialog 1   Transmission 1

bupp
entering goal/7
0.660848
15        39
          Parse 1
s(phrase(np(ng(n(noun(aircraft))),
            rnp(pp(prep(with),
                   np(ng(n(noun(AIM-9R)),
                         ng(n(noun(missile))))))))).

Creating functional parses from parse trees.

   Sample 1   Dialog 1   Transmission 1

"utflag" is turned off.

gen queue is:
     noun-gobj-noun(1)-phys_obj- #
     noun-pobj-noun(3)-phys_obj- #
     adj-noun-noun(2)-missile- # -noun-noun(3)

Finished processing fp queue: gen

first queue is:

Finished processing fp queue: first

am queue is:
     pp-with- # - # -noun(1)-noun(3)

Finished processing fp queue: am
          Sentence 1

   Sample 1   Dialog 1   Transmission 1   Sentence 1

PREP                         =  with(noun(1),noun(3))
ADJS-NOUN                    =  missile(noun(3),AIM-9R)
POBJ                         =  inst(noun(3),missile)
GOBJ                         =  inst(noun(1),aircraft)

% **********  Basic Case Generation  **********

Pass 1: Working on GOBJ = inst(noun(query-1-1),aircraft)
Pass 1: Working on POBJ = inst(noun(query-1-3),missile)
Pass 1: Working on ADJS-NOUN = missile(noun(query-1-3),AIM-9R)
```

```
Pass 1: Working on PREP = with(noun(query-1-1),noun(query-1-3))
Caching PREP is with(noun(query-1-1),noun(query-1-3)) for second pass.
Pass 2: Working on PREP = with(noun(query-1-1),noun(query-1-3))

lf(attribute(noun(query-1-3),part_of(noun(query-1-1)))).
lf(inst(noun(query-1-3),AIM-9R)).
lf(inst(noun(query-1-1),aircraft)).

% **********  Case Transformation  **********

lf(location(noun(query-1-3),on(noun(query-1-1)))).
lf(inst(noun(query-1-3),AIM-9R)).
lf(inst(noun(query-1-1),aircraft)).

% **********  Case Inference  **********

lf(location(noun(query-1-3),on(noun(query-1-1)))).
lf(inst(noun(query-1-3),AIM-9R)).
lf(inst(noun(query-1-1),aircraft)).

%PARSE_STATS - Caption query for 28 chars took 5.550 sec.

******  THRESHOLD Settings  ******

Number of KeyWords = 2

Number of Query Expressions = 3
Coarse Grain Search Threshold = 2
 Fine  Grain Search Threshold = 3

% ***************  Coarse Search  ***************

Collecting file aircraft
Collecting file Goshawk
Collecting file V/STOL aircraft
Collecting file attack aircraft
Collecting file snow aircraft
Collecting file transport aircraft
Collecting file A-3
Collecting file A-3B
Collecting file A-5C
Collecting file A-6A
Collecting file A-6B
Collecting file A-6E
Collecting file A-7
Collecting file A-7B/E
Collecting file A-7C
Collecting file A-7E
Collecting file AV-8A
Collecting file AV-8B
Collecting file B-2
Collecting file F-14
Collecting file F-14A
Collecting file F-3D
Collecting file F-3H
Collecting file F-4B
Collecting file F-4G
Collecting file F-4J
Collecting file F/A-18A
Collecting file F/A-18C
Collecting file Harrier
```

```
Collecting file QF-4
Collecting file QF-4B
Collecting file QF-4H-1
Collecting file QF-86
Collecting file QF-9
Collecting file QF-9F
Collecting file S-2
Collecting file Skyhawk 2
Collecting file T-45A
Collecting file bomber aircraft
Collecting file cargo aircraft
Collecting file fighter aircraft
Collecting file rotary wing aircraft
Collecting file A-4
Collecting file A-5
Collecting file A-6
Collecting file AC-130
Collecting file ATB
Collecting file AV-8
Collecting file C-130
Collecting file F-15
Collecting file F-3
Collecting file F-3D-1
Collecting file F-3H-1
Collecting file F-86
Collecting file F-9
Collecting file Harrier
Collecting file Hornet
Collecting file RA-5C
Collecting file S-2A
Collecting file Skyhawk
Collecting file Skywarrior
Collecting file Tomcat
Collecting file Wild Weasel
Collecting file helicopter
Collecting file mig aircraft
Collecting file A-4B
Collecting file A-4C
Collecting file A-4E
Collecting file A-4M
Collecting file AC-130A
Collecting file C-130A
Collecting file F-4
Collecting file F/A-18
Collecting file MIG-21
Collecting file military helicopter
Collecting file AH-1
Collecting file HH-1
Collecting file Super Cobra
Collecting file TH-1
Collecting file UH-1
Collecting file UH-2
Collecting file AH-1J
Collecting file AH-1W
Collecting file HH-1K
Collecting file TH-1L
Collecting file UH-1E
Collecting file UH-1N
Collecting file UH-2A
Collecting file AIM-9R
```

```
%SEARCH_QUEUE_LENGTH = 5.

%SENT 3: 256393
%RCVD 5: 256393
%RCVD 3: match_score(256393,3.0)
%SENT 3: 264968
%SENT 5: match_score(256393,3.0)
%RCVD 5: 264968
%RCVD 3: match_score(264968,3.0)
%SENT 3: 257055
%SENT 5: match_score(264968,3.0)
%RCVD 5: 257055
%RCVD 3: match_score(257055,3.0)
%SENT 3: 256395
%SENT 5: match_score(257055,3.0)
%RCVD 5: 256395
%RCVD 3: match_score(256395,3.0)
%SENT 3: 256394
%SENT 5: match_score(256395,3.0)
%RCVD 5: 256394
%SENT 5: match_score(256394,3.0)
%RCVD 3: match_score(256394,3.0)

%SEARCH_STATS - Query took 43.441 sec.

%SEARCH_RESULTS_LENGTH = 5.

Sent: resultsQueue(2, 3, [3.0-256393,3.0-264968,3.0-257055,3.0-256395,3.0-
256394]).
```

E. SUMMARY

In this chapter, we described the system approach that we have taken and the caption data we will be dealing with. The overall concept of the system is not new, in fact, there are many similarities to some of the systems we have seen in Chapter II. The major conceptual differences though include, first, our approach to dealing with the logical form by attempting to do full NL understanding and create a neutral logical form that does not require additional frames and scripts to be created by the system administrator. Second, our approach to searching both the registration data using SQL and the captions for like information. Lastly, the handling of captions themselves for describing events and spatial relationships within a multimedia object.

An additional difference that we discuss in the next two chapters is simplification of noun phrases to reduce the number of keyword records needed, an example being "Sidewinder AIM 9M missile." Previous systems have either handled them as an entire phrase or otherwise decomposed them into separate keywords without attempting to reduce them to simpler forms. By using the type hierarchy, we can reduce the expression to a simpler form (e.g., "AIM-9M") that can be used as the index that retains all of the capabilities as the other approaches, but reduces the number of the keyword records that need to be stored.

Finally, we will show a simple way to distribute the match processing to multiple computers to improve response time performance.

IV. THE NLP SUBSYSTEM

This chapter discusses the steps in creating by computer a logical form from an English NL caption and query.

A. INTRODUCTION

The goal of the NLP subsystem is to translate descriptive NL captions into an index for multimedia data objects and to translate NL queries into a form for matching against the index. As we said earlier, we decided on fully analyzing all of the words in the caption and query. To demonstrate this concept, a prototype NL parser was developed to be used with SQL for specifying retrieval requests to a multimedia database [Dulle 90]. Gross inefficiencies prevented us from using this parser at the NAWCWPNS, China Lake, for developing a more comprehensive system. We searched for a suitable NLP system to interface with our existing work and decided on the Database Generator (DBG) Message Understanding System [Montgomery 89] by Language Systems Inc. We chose this system for several reasons. First, the system was used to develop applications for five different military domains. Second, since our target application was also a military domain, it appeared that porting this software to our domain should not be difficult. Third, the system was developed in Prolog, which eased integration with our software. Last, the system was free because it was developed for Government use. But the major motivation in using the system was not to have to implement a new system from scratch.

The DBG program we started with was developed for understanding dialog conversations. Processing proceeded sequentially through the following six stages: *transmission segmentation, message segmentation, lexicalization, syntactic parser, functional parser, and template processor.* The first three stages are trivial. The parser is mostly syntactically driven and uses rule weights and verb case information to help it derive a neutral parse-tree representation. The functional parser uses a set of rules to derive the correct meaning from the parse tree. The records produced reflect the functional roles that the various phrases play in the sentence. The template processor takes the results from the functional parser and creates template (frame) records. A basic lexicon was provided with words used by pilots to report about specific events. A database of templates was also provided for grouping the information into event and object description structures.

From this initial system, we decided to retain the processing stages as they existed, but made the following modifications. In the lexicalization stage, we accepted certain punctuation characters instead of allowing them to be filtered out, including parenthesis, commas, colons, and hyphens. We retained almost all of the words in the original lexicon except the nouns and some verbs and retained the lexicon structure to avoid changes to the lexicon building routines. With respect to the predicates for defining a word, we made

further use of the functional parse predicates to designate canonical forms that are referenced in the type hierarchy, to categorize nouns according to their ancestors in the type hierarchy in order to handle nominal compounds, to handle synonyms, and to categorize qualitative and quantitative modifiers. Further verb case information was also added for both existing and new verbs to facilitate parsing. A limitation of the original lexicon which we did not correct was the inability to handle multiple word senses.

The changes we made to the syntactic parser are found mostly in the grammar rules. Grammar rules were added to handle the parsing of punctuation, descriptive noun phrases, coordinates, dates, and additional noun and verb phrase rules not covered in dialog conversational rules. In some cases, adjustment of weights also needed to be made to handle our captions. In doing so, we have tried to minimize rules changes so as not to jeopardize parsing of dialog conversation. A limitation of the original parser was the inability to handle semantic type checking for verb cases when the case information was provided in the lexicon. For example, if a verb could have as its direct object a physical object, there was no mechanism to check whether the direct object was indeed a physical object. The case information as it appeared in the verb related rules was only a means to discriminate between rule choices at a syntactic level. We did not fix this original limitation, but have noted it as an area of future modification.

Changes we made to the functional parser involved replacing all of the existing rules with new rules to handle *theme-oriented* phrases (the verb describes the state of the subject) as opposed to *agent-initiated* sentences (the verb indicates the subject is performing some task). We further extended these rules by retaining the lexical token identifiers to allow us to designate noun and verb instances in the type hierarchy, to connect modifiers with the appropriate nouns and verbs, and to indicate the constituents of various relationships involving prepositions and case grammar roles involving the verbs. Captions have more elaborate noun phrases the dialog sentences; hence we needed to indicate which of the nouns was the head noun in a noun phrase and which nouns were serving as adjectives for the head noun. We replaced almost all of the rules for generating the functional parse output so that the resulting output structure appears similar to the *slot-assertion notation* as described by Charniak and McDermott (1987).

The template processor has been replaced in its entirety with our own LF processor. This was based on our decision to derive a neutral LF structure based on general case grammar constructs as opposed to predefined domain-specific templates to facilitate our handling of the matching phase. Hence, all of the original predefined templates were discarded.

The only time the NLP subsystem can fail and produce an error is when it encounters a word in the input that is not in the lexicon. The lexicalization phase will then fail, display the word that could not be recognized, and the NLP subsystem will terminate. Any other time, the NLP subsystem will derive a LF from the current input query or caption. The LF may be complete or partial depending on whether a complete top-down parse tree was produced (examples appear in Section IV.C), there are functional parse

rules to handle all the constructs in the parse tree, all the noun and verb canonical forms that appear in the functional parse output records appear in the type hierarchy, and there are no unexpected functional parse records produced for which there do not exist LF rules.

Our experience in using the DBG system is positive. However, as with any NL system that attempts to understand English, it has inherent limitations. We have noted in this chapter the limitations and problems we encountered along the way in adapting the DBG system for understanding captions. Some features not implemented in the original DBG system or our own are automatic spelling and grammar correction, automatic word acquisition into the lexicon for new words encountered, and the ability to use semantic information during syntactic parsing. As part of this research, we can demonstrate that a parsing approach based on weights and syntactic case information alone is not sufficient for accurate NL processing. We recognize that the DBG version we are using may have become obsolete during the course of this research.

B. THE LEXICON

1. Lexicon Structure

The DBG lexicon consists of predicates where the functor represents the part of speech and the first argument is the English word token. Additional arguments can be used to indicate morphological structure (e.g., plural form, proper noun, etc. for nouns and past, present, past participle, etc. for verbs); functional parse features (specifying how a word should be classified and processed); functional parse categories (the semantic representation of a word); and semantic case information (for handling verb phrase structures).

The word tokens can either be an individual word (e.g., "missile") or a string of words to be treated as one token (e.g., "AIM 9R"). We have pursued a modest knowledge-intensive approach to the handling of nominal compounds as described in Gay and Croft (1990). We have also used the NASA Thesaurus [1988] as a reference in defining words in the lexicon and their corresponding concepts in the type hierarchy. The original DBG system used only lower case words in understanding dialog conversation. For written captions, we used upper case characters for proper nouns and acronyms. The original DBG system did not recognize a proper noun that was entered in lower case and this limitation has been carried into our own system. However, in the future we can incorporate a simple lookup to check for lower case proper nouns and convert them to upper case.

2. Semantic Categories

There has been much work in trying to derive canonical semantic representations that words could be mapped to. Schank (1975, 1977) and his students have had the most dominant effect in the field. He devised 11 basic verb types (ACTs) for mapping actions/events, termed Conceptual Dependency (CD)

Theory. The problem, though, is that these ACTs are in many ways too basic; to describe one verb in English may involve several ACTs and related structures. Schank later augmented them with six additional higher level social ACTs [Schank 79] when attempting to understand the Gettysburg address. The DBG philosophy itself (the basis for this parser) can also be contrasted with CD in the way event structures are organized. Recall the basic semantic problem discussed in Chapter II where Dumais (1988) proposed "latent semantic indexing." Briggs (1985) from a different perspective proposed the use of Sanskrit for as the canonical form because of the scientific and formal way in which it was written. From this, we can say that the methodology we have chosen is not based solely on one work. We have tried to incorporate these ideas with our own in designing the lexicon. For example, out of 320 verbs, we have derived 88 canonical verbs.

When entering a word definition, we have the option of specifying an alternate semantic category using the functional parse category, *fpcat*, predicate. For example, the term "AIM 9R" in the following lexicon definition

```
noun('AIM 9R', morph(4), fp(missile), fpcat('AIM-9R')).
```

has the term "AIM-9R" substituted for it when creating the logical form. If it is not provided, then the original word is assumed to be the semantic category. The functional parse, *fp*, predicate indicates that "AIM 9R" is classified as a missile within the type hierarchy. This class, however, doesn't necessarily have to be the immediate parent. The class is used primarily in the mapping routines to handle nominal compounds and subject-verb-object functionality during the functional parse to logical form transformation.

As we stated in the Introduction, a limitation of the DBG lexicon is its ability to only handle one semantic definition. The term "Sidewinder" refers to a Sidewinder missile and not the snake. Further, there is always the question of what the semantic category should be. "Harrier II" is one of those terms in a caption. Do we categorize this word as "Harrier" or leave it as is? If we choose the former then all searches for "Harrier II" will entail looking up "Harrier" as well as "Harrier II." Choosing the latter allows us to search for either term individually. However, if "Harrier II" is specified as a subclass of "Harrier," a search for "Harrier" will include a search for "Harrier II" as well. As you will discover in the next chapter, each semantic category has an associated keyword file of caption identifiers. What this decision amounts to is deciding on one large index file or two smaller index files.

Another important decision in determining the semantic categories is the treatment of verbals (e.g., "assembly," "elevation," "operation," etc.). Do we need to make a distinction between the verbals and the verb, i.e., should "assembly" and "assemble" both exist in the type hierarchy? Anick (1991) pointed out that stemming will sometimes map unrelated words to the same stem, inadvertently affecting the meaning. To illustrate the problem, we assume the lexicon has a separate entry for "assembly" and "assemble." Looking ahead to the matching, if we treat the two separately, then the phrases:

`"soldering assembly area"` and `"the area for assembling solder"`

will not match completely using the syntactic approach. The coarse-grain match will perform lookup on "soldering," "assembly," and "area" as well as "assembling" and "solder." "Soldering" does not match "solder." Likewise, the root form of "assembling" ("assemble") does not match "assembly." Only "area" would match and contribute towards deriving the match score (specifically, an increment of 1).

On the other hand, if we used only roots, then the two sentences would have "soldering" being treated as "solder" and thus matching "solder" in the second sentence. "Assembly" is treated as "assemble" and is matched to the root of "assembling" in the second sentence. "Area" matches "area." There are more matches possible (increment of 3) allowing progression to the fine-grain match for further analysis.

Consider current keyword lookup strategies where a searcher can specify: "assembl*," "solder*," and "area." This search would indeed find both of the previous two strings (match score increment of 3). The difference being that there would exist multiple "assembl*" files as well as multiple "solder*" files. In our latter case, we would have just one "assemble" file and one "solder" file. Our match approach should at least be comparable to a keyword lookup using pattern matching characters to handle the different endings; this would seem to argue for using root forms. However, we desire more than just simple pattern matching. We need to be able to differentiate between the event represented by a verbal and the root noun itself. We also need to distinguish between when it is used as a verb and when is used as a noun. There is no set solution to the treatment of verbals (or PP-modifiers) as indicated by Allan (1987). We hypothesize that we treat the two words separately and establish an explicit link between them; we discuss how this can be done when we discuss the type hierarchy. Appendix C contains a complete listing of the lexicon.

C. SYNTACTIC PARSING

The syntactic parser attempts to derive a top-down parse of the sentence using parse rules with associated weights. Should the top-down parse fail, a bottom-up parser is invoked to create those component structures that it recognizes. For all the captions and queries we have tested, we have been able to derive a top-down parse for each. The output of the syntactic parser is a parse tree that corresponds to the sentence and has the highest weights when rules are combined. Some semantic verb case information can be used to prefer one rule over another, but their effectiveness is limited; we discuss these limitations later in this section. Once the parse tree is produced, there is no backtracking into the syntactic parser to arrive at a different parse tree. It is assumed that the parse tree is neutral enough for deriving the semantic meaning using the functional parse rules.

With the DBG system, deriving a neutral parse tree rests entirely on the accuracy of the rule weights. If a correct neutral parse tree is always generated, then writing the appropriate semantic interpretation rules based on the tree structure is a relatively simple task. However, we have encountered cases where adding additional phrases to the end of a sentence has resulted in a different tree structure for the first part of the

sentence. Adding a determiner before an apparent noun has also given us a different tree, not just for the noun phrase, but for the entire caption. The parser has also produced erroneous parse trees with words that can be treated as either verbs or nouns. Likewise, the existence or absence of adjective and adverb modifiers have also generated entirely different caption structures. We have documented many of these problems in this section. Writing the functional parse rules to derive the semantic meaning based on the parse tree has resulted in some amount of code duplication and generalities being made. We believe that a more accurate parse tree and interpretation may be obtained in the future if the functional parse rules were used in conjunction with the syntactic parse rules during parsing of the sentence. Otherwise, we need some way to have the functional parse rules fail and backtrack into the syntactic parser to look at alternative parse trees. A listing of the grammar used by the syntactic parser can be found in Appendix D.

1. Additions

The parsing of captions can be relatively straightforward in most cases, while in other cases the task is not as easy. In this section, we discuss some of the simpler additions and modifications that were made to the parse rules and captions to handle non-ordinary structures such as nominal compounds, commas, coordinates, dates, and an ordered sequence of noun phrases. The next section discusses the complications that arose.

a. Noun Phrases

We have observed that captions, at least those written at the Center, are dominated mostly by noun phrases and that these noun phrases are very descriptive. It is not very often that you will hear someone speak "United States Navy F A 18 A bureau number 164235 Hornet aircraft." The original parser handled simple noun groups such as lexically-defined adjectives preceding the noun, a noun followed by an integer, compound nouns, etc. We extended the rules to handle numbers other than integers (e.g., "#E"), letter identifiers, and appositives; the weights then needed to be adjusted accordingly.

In Eg. 4.1, the underlined words form the noun phrase and appositive. Note that the majority of words are nouns acting as adjectives and that the head noun for the second noun phrase is "aircraft." The biggest problem we had in parsing the appositive was when we used comma's instead of the parenthesis. This problem will be addressed shortly.

b. Commas

The original parser did not have any rules for handling commas, as commas are not recognized in spoken English. Nonetheless, we had captions with commas that had to be dealt with. In Eg. 4.2, the first caption has the comma behaving like the preposition "with." In the second caption, it is an appositive to the missile (EDM-4 specifies the engine design model) while in the third caption, it behaves like the preposition "of." In the next three, the comma is used to delimit appositives; while in the last case,

Example 4.1. Detailed Noun Phrase from Caption 163030.

```
FAE weapons on USMC VMA-513 AV-8A BU# 158389 Harrier aircraft (nose 6 and
WF on tail).

s(phrase(np(ng(n(noun(FAE)),
                ng(n(noun(weapons)))),
             rnp(pp(prep(on),
                    np(ng(n(noun(USMC)),
                          ng(n(noun(VMA-513)),
                             ng(n(noun(AV-8A)),
                                ng(n(noun(BU#)),
                                   num(integer(158389)),
                                   ng(n(noun(Harrier)),
                                      ng(n(noun(aircraft)),
                                         app(lparen((),
                                             np(cnp(ng(n(noun(nose)),
                                                       num(integer(6)))),
                                                conj(and),
                                                cnp(ng(n(noun(WF))),
                                                    rnp(pp(prep(on),
                                                           np(ng(n(noun(tail))))
)))),
                                             rparen())))))))))))))).
```

it is used to delimit noun phrases in a sequence of conjunctive/disjunctive noun phrases. To handle these various situations, parse rules in (1), (2), and (3) were added. The first rule was used to handle trailing noun phrases preceded by a comma. The second set of rules handled appositives, and the last set handled the sequence of noun phrases.

Example 4.2. Some captions containing commas.

ID	CAPTION
110169	Bat and Standard Arm missiles with A-6A aircraft, nose 562 and NG on tail.
216383	... with Phoenix missile XAIM-54C, EDM-4.
262865	Sidewinder AIM 9R missile on stand, front view.
10862	Sparrow III, Guardian of the Skies, operational with Seventh Fleet.
161082	air to air view of Sidewinder LAIM, AIM 9L, launch from helicopter.
218997	LWIR, Long Wavelength Infrared, side view.
219907	Harm AGM-88, USAF IR Maverick AGM-65D, ALG-119 ECM pod, Standard Arm, and Shrike AGM-45.

```
np:0.81 ==> ng, comma, np.                                          (1)

ng:0.98 ==> n, app, ng.                                             (2)
ng:0.91 ==> n, app.
app ==> comma, np, comma.

np:0.71 ==> cnp, comma, cnp, comma, conj, cnp.                      (3)
cnp ==> ng, rnp.
rnp ==> pp.
pp ==> bprep, cnp, comma, cnp, comma, conj, cnp.
```

c. Coordinates

Some of the captions encountered explicitly stated coordinate positions. We have chosen to deal with a coordinate as one indivisible unit by using the parse rule shown in (4). Additional processing of

```
ng:0.84 ==> n.                                                          (4)
n ==> coordinate.
coordinate ==> integer, quote, dir, by, integer, quote, dir.
```

the coordinates can be done during the time the logical form is being created if the need should arise. Eg. 4.3 shows the result of applying this parse rule to the third sentence of Caption 180657.

Example 4.3. Coordinates from Caption 180657.

```
synchro firing at 4505'N x 34' E from camera 44.

s(phrase(np(ng(adjp(adj(synchro)),
               ng(n(noun(firing)))),
            rnp(pp(prep(at),
                   np(ng(n(coordinate(integer(4505),
                                      quote('),
                                      dir(N),
                                      by(x),
                                      integer(34),
                                      quote('),
                                      dir(E)))),
                      rnp(pp(prep(from),
                             np(ng(n(noun(camera)),
                                   num(integer(44)))))))))))).
```

d. Dates

The parsing of dates is handled similarly to coordinates, i.e., we deal with it as one indivisible unit. The parse rule is shown in (5). Additional processing of the date can also be done during the time the logical form is being created. Eg. 4.4 provides a date example.

```
ng:0.84 ==> n.                                                          (5)
n ==> date.
date ==> month, day, comma, year.
```

Example 4.4. Date from Caption 257055.

```
intercept with QF-86 drone on April 12, 1989.

s(phrase(np(ng(n(noun(intercept))),
            rnp(pp(prep(with),
                   np(ng(n(noun(QF-86))),
                      rnp(pp(prep(on),
                             np(ng(n(date(month(April),
                                           day(integer(12)),
                                           comma(,),
                                           year(integer(1989))))))))))))).
```

e. *Ordered Sequence*

A different type of sequence problem involved an explicit ordering being stated beforehand, such as "top-down," "bottom-up," "left-to-right," etc. To handle this situation, we artificially inserted a colon into the caption and defined specific ordering relations like the ones previously mentioned. In Eg. 4.5,

Example 4.5. Ordered sequence in Caption 219907.

```
air to air view of Air Force USAF# 69263 F-4G Wild Weasel with weapons
loaded top to bottom: Harm AGM-88, USAF IR Maverick AGM-65D, ALG-119 ECM
pod, Standard Arm, and Shrike AGM-45.

   ordseq(order(top to bottom),
          colon(:),
          seq(cnp(ng(n(noun(Harm)),
                      ng(n(noun(AGM-88))))),
              comma(,),
              seq(cnp(ng(n(noun(USAF)),
                          ng(n(noun(IR)),
                             ng(n(noun(Maverick)),
                                ng(n(noun(AGM-65D))))))),
                  comma(,),
                  seq(cnp(ng(n(noun(ALQ-119)),
                              ng(n(noun(ECM pod))))),
                      comma(,),
                      seq(cnp(n(noun(Standard Arm))),
                          comma(,),
                          conj(and),
                          cnp(ng(n(noun(Shrike)),
                                 ng(n(noun(AGM-45)))))))))))).
```

the phrase after the colon is nothing more that a noun phrase sequence. The ordering relation allows us to indicate what is next to next to what (using the *seq* identifier) and in what order.

2. Complications

We now discuss some of the parsing complications that have arisen using the syntactic parser. One of our goals in the parsing was to minimize changes to the captions themselves. In some cases, however, we had no choice. Fixes were made accordingly by either adding or modifying the parse rules and/or modifying the caption by hand. For other cases there was nothing we could do because we felt there was a need to make the syntactic parser more semantic driven; either by combining the functional parse rules with the syntactic parse rules or allowing the functional parser to backtrack into the syntactic parser for alternatives, both of which are not presently implemented because of our desire to minimize changes until we better understood the parsing problems. Some of the captions shown in the examples are the original captions; others are shown in various stages of modification.

a. *Objects of Prepositions*

The parsing of prepositions is handled by treating the preposition as a relation with preceding and succeeding noun objects as the arguments to the relationship. E.g. 4.6(a) presents a case where the

preceding object is listed once - at the beginning of the sequence. This usage was easy for a human to grasp, but not so easy for the parser. On attempting to parse this sentence, the top-down parse failed, resulting in the bottom-up parse shown in the example. Note that the first preposition "on" was treated as an adverb and

Example 4.6. Problem with Object of Preposition in Caption 234756.

(a) Original Caption.

```
drone painted silver with red on nose, under wings, on tail, and on fin.

np(ng(n(noun(drone))),
   rnp(ajc(ppt(pastpart(painted)),
           rvp(np(ng(n(noun(silver))),
                  rnp(pp(prep(with),
                         np(ng(n(noun(red))),
                            rnp(am(avp(adv(on))))))))))),
np(ng(n(noun(nose)))),
am(avp(pp(prep(under),
          np(ng(n(noun(wings))))))),
am(avp(pp(prep(on),
          np(ng(n(noun(tail))))))),
am(avp(pp(prep(on),
          np(ng(n(noun(fin))))))).
```

(b) Modified Caption.

```
drone painted silver with red on nose, red under wings, red on tail, and
red on fin.
```

the remainder of the sequence appeared as individual parse trees. We chose to handle this problem by modifying the caption. We introduced the preceding object ("red") for each of the prepositions in the sequence as shown in E.g. 4.6(b).

b. Commas

Section IV.C.1.B described the approach that we took to handle commas in the captions. Inconsistent use of commas in the captions and the problems with these rules motivated our decision to simplify the handling of appositives and commas in general. Had we not done so and retained rule (1), the parser working on the caption in Eg. 4.6(b) produced an erroneous parse tree (see Eg. 4.7(a)). Examine the portion of the parse tree which contains "red under wings, red on tail." It is being embedded within one of the prepositional phrases in the sequence. When rule (1) was removed, the parser generated the correct parse structure shown in Eg. 4.7(b).

Note however that even with rule (1) removed, we needed to define "with" as a *brep* in the lexicon. If it were not defined as such, the top-down parser would fail and the bottom-up parser would attempt to parse what it could (see E.g. 4.7(c)). The problem with generalizing the prepositional phrase rule (*pp)* not to depend on a *bprep* but just a regular *prep* was that "with" behaves like a conjunction and not all

prepositions are conjunctions (eg., the use of a conjunction ("and") instead of "with" didn't make sense when reading the caption). Our solution required the use of commas only when they are used to connect a

Example 4.7. Parse trees for Caption 234756.

(a) Top-down parse with Rule (1) -

```
s(phrase(np(ng(n(noun(drone)))),
         ajc(ppt(pastpart(painted)),
             rvp(np(ng(n(noun(silver))),
                    rnp(pp(bprep(with),
                           cnp(ng(n(noun(red))),
                               rnp(pp(prep(on),
                                      np(ng(n(noun(nose))))))),
                           comma(,),
                           cnp(ng(n(noun(red))),
                               rnp(pp(prep(under),
                                      np(ng(n(noun(wings))),
                                         comma(,),
                                         np(ng(n(noun(red))),
                                            rnp(pp(prep(on),
                                                   np(ng(n(noun(tail))))))))))),
                           comma(,),
                           conj(and),
                           cnp(ng(n(noun(red))),
                               rnp(pp(prep(on),
                                      np(ng(n(noun(fin))))))))))))))).
```

(b) Top-down parse without Rule (1) -

```
s(phrase(np(ng(n(noun(drone)))),
         ajc(ppt(pastpart(painted)),
             rvp(np(ng(n(noun(silver))),
                    rnp(pp(bprep(with),
                           cnp(ng(n(noun(red))),
                               rnp(pp(prep(on),
                                      np(ng(n(noun(nose))))))),
                           comma(,),
                           cnp(ng(n(noun(red))),
                               rnp(pp(prep(under),
                                      np(ng(n(noun(wings))))))),
                           comma(,),
                           cnp(ng(n(noun(red))),
                               rnp(pp(prep(on),
                                      np(ng(n(noun(tail))))))),
                           comma(,),
                           conj(and),
                           cnp(ng(n(noun(red))),
                               rnp(pp(prep(on),
                                      np(ng(n(noun(fin))))))))))))))).
```

(c) Bottom-up parse without *bprep* rule -

```
np(ng(n(noun(drone))),
   rnp(ajc(ppt(pastpart(painted)),
           rvp(np(ng(n(noun(silver))),
                  rnp(pp(prep(with),
                         np(ng(n(noun(red))),
```

```
                                    rnp(am(avp(adv(on))))))))))),
np(ng(n(noun(nose)))),
np(ng(n(noun(red))),
   rnp(am(avp(adv(under))))),
np(cnp(n(noun(wings))),
   comma(,),
   cnp(ng(n(noun(red))),
       rnp(pp(prep(on),
              np(ng(n(noun(tail))))))),
   comma(,),
   conj(and),
   cnp(ng(n(noun(red))),
       rnp(pp(prep(on),
              np(ng(n(noun(fin)))))))).
```

sequence of noun phrases with a final conjunction. The use of commas to handle acronyms, identifiers, "with," and "of" phrases needed to be generalized into either an appositive form by introducing the parenthesis into the captions, or rewriting the caption without the noun phrase being offset by the comma.

c. Embedded Sentences

When using the parser, it is possible for a phrase/sentence to have multiple parse structures depending on where it appears in a caption. Eg. 4.8(a) show the parse tree for the underlined portion of Caption 258795. If we now parse the entire sentence, we arrive at the parse tree show in 4.8(b). Note the differences for the same underlined portion.

Example 4.8. Parse trees for Caption 258795.

`Cdr Antonio in cockpit with` <u>`night`</u> <u>`vision`</u> <u>`goggles`</u> <u>`turned`</u> <u>`on`</u>`.`

(a) Underlined portion of Caption 258795.

```
s(phrase(np(ng(n(noun(night vision)),
               ng(n(noun(goggles))))),
         ajc(vnom-*vtype(operate)*(ppt(pastpart(turned)),
                  part(on))))).
```

(b) Entire sentence of Caption 258795.

```
s(phrase(np(ng(n(noun(Cdr)),
               ng(n(noun(Antonio)))),
            rnp(pp(prep(in),
                   np(ng(n(noun(cockpit))),
                      rnp(am(avp(pp(prep(with),
                                    np(ng(n(noun(night vision)),
                                          ng(n(noun(goggles)))),
                                       rnp(ajc(ppt(pastpart(turned)))))),
                                           adv(on))))))))).
```

We surmise that the multiple parse structure problem arises because we do not have the higher level rules embedded in the lower level rules. But to do so would introduce infinite recursion into the rules. Our best viable solution is to recognize similar patterns in the two structures in the functional parse

and semantic analysis phase and attempt to generate a logical form based on the similarities, realizing that imperfections will exist.

A slightly different problem is when we have a prepositional phrase encompassing two sentences connected by a conjunction. In Eg 4.9, we have a problem with distributing "view of" to both sentences. Notice that "view of" only applies to the "bomb" and that we have no obvious way of applying

Example 4.9. Parse tree for Caption 178010.

```
view of bomb wrapped and wires attached.

s(declarative(cs(np(ng(n(noun(view))),
                    rnp(ofp(of(of),
                            np(ng(n(noun(bomb))))))),
                 vp(v(past(wrapped)))),
              conj(and),
              cs(np(ng(n(noun(wires)))),
                 vp(v(past(attached)))))).
```

this to the second sentence. Further examination would seem to suggest "wrapped" and "attached" as adjectives. However, even creating both terms as adjectives, the rules still favor the *np*, *vp* form as seen above. Adding a secondary rule like (6) does not solve the problem since the rule from the parse tree would

```
declarative ==> np, cs, conj, cs.                                    (6)
```

always subsume this rule. The object of the "attach" is also a problem as we must look back into the history and find the last noun that can have something attached. Looking solely for the last noun would fail as can be seen in the sentence "view of bomb on skid and wires attached." Hence, we need to explicitly state in the caption what the wires are attached to.

Other captions showing similar problems are shown in Eg. 4.10(a). To resolve our problem with the conjunction and implicit objects, we have split the sentence into two and explicitly specified the objects as shown in Eg. 4.10(b).

Example 4.10. Original and Modified Captions having Embedded Sentences.

(a) Original Captions.

ID	CAPTION
180657	dummy just leaving cockpit with seat rockets burning.
181761	interior damage and exterior view of boxcar at bldg 63.
182713	Harpoon missile launch aboard USS High Point PCH-1 hydrofoil underway in Puget sound and firing one harpoon.

(b) Possible Modifications.

ID	CAPTION
258795	Cdr Antonio in cockpit. night vision goggles turned on.
178010	view of bomb wrapped. view of wires attached to bomb.
180657	dummy just leaving cockpit. seat rockets burning.

```
181761     interior damage to boxcar. exterior view of boxcar at bldg 63.
182713     Harpoon missile launch aboard USS High Point PCH-1 hydrofoil
           underway in Puget sound. hydrofoil firing one harpoon.
```

d. Conjunctive Noun Phrases

The first case we examine has a preposition preceding two noun phrases connected via a conjunction as shown in Eg. 4.11. Applying some domain knowledge, we can assume that the "Standard Missile II-N" is part of the aircraft and that the "EX-62 TDD" is in the nose of the missile. The caption is ambiguous in that "nose" could also refer to the aircraft. The parse tree produced, however, is correct; handling the "nose" issue is an anaphoric reference problem. If we now maintain the same caption structure but have different words in the caption, then the parse tree is inappropriate.

Example 4.11. Parse tree for Caption 238225.

Skywarrior aircraft <u>with</u> Standard Missile II-N <u>and</u> EX-62 TDD in nose.

```
s(phrase(np(ng(n(noun(Skywarrior)),
                ng(n(noun(aircraft)))),
            rnp(pp(prep(with),
                   cnp(n(noun(Standard Missile II-N))),
                   conj(and),
                   cnp(ng(n(noun(EX-62)),
                          ng(n(noun(TDD)))),
                       rnp(pp(prep(in),
                              np(ng(n(noun(nose))))))))))).
```

In Eg. 4.12, the underlined portion is structurally similar to Caption 238225. We naturally assume that "plume" and "aircraft" are not part of the aircraft. In fact, "missile away from aircraft" and "plume and aircraft in view" are two separate thoughts and need to be treated that way. The parser, however, sees little difference between this caption and the one in Eg. 4.11 and arrives at the same parse tree structure. Furthermore, if we now attempt to parse the entire caption, we get an even more erroneous interpretation (see Eg. 4.12(b)). In this parse tree, we have that "aircraft with plume" and "aircraft in view"

Example 4.12. Parse trees for Caption 241950.

<u>missile</u> away from aircraft <u>with</u> <u>plume</u> <u>and</u> <u>aircraft</u> <u>in</u> <u>view</u>.

(a) Underlined portion of caption.

```
s(phrase(np(ng(n(noun(missile))),
            rnp(pp(prep(with),
                   cnp(n(noun(plume))),
                   conj(and),
                   cnp(ng(n(noun(aircraft))),
                       rnp(pp(prep(in),
                              np(ng(n(noun(view))))))))))).
```

(b) Entire sentence of caption.

```
s(phrase(np(ng(n(noun(missile))),
            rnp(am(avp(adv(away),
                       pp(prep(from),
                          cnp(ng(n(noun(aircraft))),
                              rnp(pp(prep(with),
                                     np(ng(n(noun(plume))))))),
                          conj(and),
                          cnp(ng(n(noun(aircraft))),
                              rnp(pp(prep(in),
                                     np(ng(n(noun(view))))))))))))).
```

(c) Caption 241950 rewritten as two sentences.

```
missile is away from aircraft. plume and aircraft are in view.

s(declarative(np(ng(n(noun(missile)))),
              vp(v(be(is)),
                 rvp(am(avp(adv(away),
                            pp(prep(from),
                               np(ng(n(noun(aircraft))))))))))).

s(declarative(np(cnp(n(noun(plume))),
                 conj(and),
                 cnp(n(noun(aircraft)))),
              vp(v(be(are)),
                 rvp(am(avp(pp(prep(in),
                               np(ng(n(noun(view))))))))))).
```

are both objects of "from." The way the parse tree reads then is that the missile is away from some aircraft that has a plume and some other aircraft that is in view.

As we have seen, the parse rules with their associated weights cannot handle the semantic differences. Hence, we need to manually rephrase the caption to remove the ambiguity. The easiest solution is to replace "with" with a period and create two sentences. In the subsequent attempt to reparse this caption, the parser again produced incorrect interpretations. The reason is that "away" in the first sentence is an adverb and the parser has no verb with which to associate it. In the new second sentence, the parser would associate "in view" with "aircraft" only, not with the conjunction of "plume" and "aircraft." Upon examining the parse rules and testing various deviations in the caption, we discovered that we must manually introduce "be" verbs into both phrases to get the parser to generate the correct interpretation. The resulting caption and parse trees are shown in Eg. 4.12(c).

Another conjunctive noun phrase problem is shown in Eg. 4.13(a). Here we naturally assume that both round 3 and round 4 together are on the centerline station. The parser, however, associates the preposition with the second noun only. Eg. 4.13(b) shows a similar situation. From this interpretation, only "round 4" is on the "centerline station" and only "radar" is "in front of" the "target." The problem stems from the existence of rule (7) and no rule (8) below. Introducing such a rule, however, would lead to incorrect parse structures for those captions in which the prepositional phrase following the second noun in

Example 4.13. Examples of Conjunctive Noun phrase problems.

(a) From the second sentence of Caption 34070.

```
rounds 3 and 4 on centerline station with multiple bomb rack configuration.

s(phrase(np(cnp(ng(n(noun(rounds)),
                   num(integer(3)))),
            conj(and),
            cnp(ng(n(callsign(integer(4)))),
                rnp(pp(prep(on),
                       np(ng(n(noun(centerline station))))))))))).
```

(b) From the second sentence of Caption 241426.

```
tank and radar in front of target.

s(phrase(np(cnp(n(noun(tank))),
            conj(and),
            cnp(ng(n(noun(radar))),
                rnp(pp(prep(in front of),
                       np(ng(n(noun(target)))))))))).
```

(c) Rephrasing of underlined phrase from (a) using a "be" verb.

```
round 3 and round 4 are on centerline station.

s(phrase(np(cnp(ng(n(noun(round)),
                   num(integer(3)))),
            conj(and),
            cnp(ng(n(noun(round))),
                rnp(rc(np(ng(n(callsign(integer(4))))),
                       vp(v(be(are)),
                          rvp(am(avp(pp(prep(on),
                                        np(ng(n(noun(centerline station))))))
))))))))).
```

(d) Rephrasing of (b) using a "be" verb.

```
tank and radar are in front of target.

s(declarative(np(cnp(n(noun(tank))),
                 conj(and),
                 cnp(n(noun(radar)))),
              vp(v(be(are)),
                 rvp(am(avp(pp(prep(in front of),
                                np(ng(n(noun(target))))))))))).
```

the conjunction actually applies to the second noun. For example, we would have a problem with the caption "tank and radar on the van" where only the radar is on the van. Adjusting weights for each of the two rules would not help, when what is really needed is to apply semantic information.

```
np:0.90 ==> ng, rnp.                                                   (7)

phrase ==> np, rnp.                                                    (8)
```

A plausible solution to Eg. 4.13(a) would be to duplicate the "round" noun and include it for "4" and introduce a "be" verb before the preposition (see Eg. 4.13(c)). However, even this modification does not help. We now have a problem that "round" and "4" are being separated into two parse structures and that the verb is associated only with the term "4" instead of with both. Adding a "be" verb to Eg. 4.13(b)'s caption however, results in a correct parse tree (see Eg. 4.13(d)).

In certain captions however, there is nothing we can do because we can neither insert a word nor split the sentence naturally into two without excessive duplication. In Eg. 4.14(a), if we used the same strategy as before, we would attempt to insert a "be" verb before the preposition. In this case, the caption doesn't read well. If we try to augment it with a "wh" word (eg., "which"), we just complicate the problem,

Example 4.14. Conjunctive Noun phrase problem in Caption 168578.

(a) From the second sentence of Caption 168578.

```
closeup view of Walleye II bomb and data link pod on A-7E aircraft.

s(phrase(np(detg(det(quant(closeup))),
            ng(n(noun(view))),
            rnp(ofp(of(of),
                    np(cnp(ng(n(noun(Walleye II)),
                              ng(n(noun(bomb))))),
                       conj(and),
                       cnp(ng(n(noun(data link pod))),
                           rnp(pp(prep(on),
                                  np(ng(n(noun(A-7E)),
                                        ng(n(noun(aircraft))))))))))))).
```

(b) Inserting a "be" verb into the caption in (a).

```
closeup view of Walleye II bomb and data link pod which are on A-7E aircraft.

s(phrase(np(detg(det(quant(closeup))),
            ng(n(noun(view))),
            rnp(ofp(of(of),
                    np(cnp(ng(n(noun(Walleye II)),
                              ng(n(noun(bomb))))),
                       conj(and),
                       cnp(ng(n(noun(data link pod))),
                           rnp(rc(relpro(which),
                                  vp(v(be(are)),
                                     rvp(am(avp(pp(prep(on),
                                                   np(ng(n(noun(A-7E)),
                                                         ng(n(noun(aircraft)))
))))))))))))))))).
```

as Eg. 4.14(b) shows. There is no other obvious place to insert a verb, and there is no easy way to split the sentence short of copying each word in the sentence (e.g., "closeup view of Walleye II bomb. closeup view of data link pod on A-7E aircraft."). Unfortunately, this latter strategy turned out to be our best solution for this and Caption 241426 (from Eg. 4.13(b)).

Some problems can easily be solved by performing a simple transposition in the sentence. In Eg. 4.15(a), notice the "with" problem in the first appositive. The conjunction ties together two separate noun phrases and these two noun phrases are the object of the preposition "with." Any analysis would result in "nose" having a night hawk logo as well as a tail. Any rule changes introduced would cause erroneous results for some of the previous captions. Without an obvious parsing solution to this problem, we resort to transposing the noun phrases in the conjunction to give us a more accurate parse (see Eg. 4.15(b)).

Example 4.15. Conjunctive Noun phrase problem in Caption 257019.

(a) From the second sentence of Caption 168578.

```
AV-8B night attack BU# 162966 aircraft (nose 87 with night hawk logo and
tail 162966) from MAD (Marine Air Detachment).

ng(n(noun(aircraft)),
   app(lparen((),
       np(ng(n(noun(nose)),
             num(integer(87))),
          rnp(pp(prep(with),
                 cnp(ng(n(noun(night hawk)),
                        ng(n(noun(logo))))),
                 conj(and),
                 cnp(ng(n(noun(tail)),
                        num(integer(162966))))))),
       rparen())))
```

(b) Transposition of words in appositive from (a).

```
AV-8B night attack BU# 162966 aircraft (tail 162966 and nose 87 with night
hawk logo) from MAD (Marine Air Detachment).

ng(n(noun(aircraft)),
   app(lparen((),
       np(cnp(ng(n(noun(tail)),
                 num(integer(162966)))),
          conj(and),
          cnp(ng(n(noun(nose)),
                 num(integer(87))),
              rnp(pp(prep(with),
                     np(ng(n(noun(night hawk)),
                           ng(n(noun(logo))))))))),
       rparen()))))).
```

We have had to solve our problem artificially by forcing the caption to be written a certain way. We have often asked ourselves in using the DBG syntactic parser whether the existence of the multitude of parse rules and their associated weights is detrimental to arriving at a likely parse solution.

e. Conjunctive Verb Phrases

As with the previous section, we have encountered problems when dealing with conjunctive verb phrases. In Eg. 4.16(a), the top-down parser failed when trying to create the parse tree using the original DBG parse rules. We decided to approach this problem as if we were dealing with two sentences. The first problem deals with "drogue chute above." After examining the caption in more detail and consulting with the Photo Lab, we concluded that the parachute was above the dummy. Thus, instead of creating rules to handle the caption as it was written, we decided to manually modify the caption by specifically introducing "is above dummy" into the first half of the caption (see Eg. 4.16(b)). With the first part correct, we attempted to parse the entire caption (see Eg. 4.16(c)). Note that by adding the second verb phrase, a different parse rule was applied for the first part. However, this parse tree is inaccurate. Believing the problem to be English in the second part of the caption as well, we introduced a "be" verb into the second half as shown in Eg. 4.16(d).

The parse structure changed slightly in Eg. 4.16(d), but it is still incorrect. Upon inspecting the caption closer and looking at the existing parse rules, we saw that the sentence did not conform to any existing parse rules. Analyzing the sentence, we came up with parse rule (9) in addition to (10).

Example 4.16. Conjunctive Verb phrase problem in second sentence of Caption 237492.

(a) Bottom-up parse results after top-down parse failed.

```
drogue chute above but not opened.

np(ng(n(noun(drogue)),
      ng(n(noun(chute))))),
am(avp(av(avh(neg(not))))),
vp(v(past(opened))).
```

(b) Successful parse on rewritten first half of caption.

```
drogue chute is above dummy.

s(declarative(np(ng(n(noun(drogue)),
                    ng(n(noun(chute))))),
              vp(v(be(is)),
                 rvp(am(avp(pp(prep(above),
                              np(ng(n(noun(dummy)))))))))))).
```

(c) Inaccurate parse when combining rewritten first half with second half.

```
drogue chute is above dummy but not opened.

s(phrase(np(ng(n(noun(drogue)))),
         rnp(rc(np(ng(n(noun(chute)))),
                vp(v(be(is)),
                   crvp(am(avp(pp(prep(above),
                                  np(ng(n(noun(dummy)))))))),
                   conj(but),
                   crvp(am(avp(av(avh(neg(not))))))))),
      ajc(ppt(pastpart(opened))))).
```

(d) Inaccurate parse with both rewritten halves.

```
drogue chute is above dummy but is not opened.

s(phrase(np(ng(n(noun(drogue))),
            rnp(rc(np(ng(n(noun(chute)))),
                   vp(cvp(v(be(is)),
                          rvp(am(avp(pp(prep(above),
                                        np(ng(n(noun(dummy))))))))),
                      conj(but),
                      cvp(v(be(is)),
                          rvp(am(avp(av(avh(neg(not)))))))))),
         ajc(ppt(pastpart(opened))))).
```

(e) Inaccurate parse with rewritten first half and parse rule (9).

```
drogue chute is above dummy but not opened.

s(phrase(np(ng(n(noun(drogue))),
            rnp(rc(np(ng(n(noun(chute)))),
                   vp(v(be(is)),
                      crvp(am(avp(pp(prep(above),
                                     np(ng(n(noun(dummy)))))))),
                      conj(but),
                      crvp(am(avp(av(avh(neg(not))))))))),
         ajc(ppt(pastpart(opened))))).
```

(f) Accurate parse with rewritten halves and parse rule (9).

```
drogue chute is above dummy but not opened.

s(declarative(np(ng(n(noun(drogue)),
                    ng(n(noun(chute))))),
              vp(v(be(is)),
                 rvp(am(avp(pp(prep(above),
                               np(ng(n(noun(dummy))))))))),
              conj(but),
              vp(v(auxp(be(is),
                        av(avh(neg(not)))),
                   vf(ppt(pastpart(opened)))))).
```

```
declarative ==> np, vp, conj, vp.                              (9)

declarative ==> np, vp.                                        (10)
```

With this rule added, we parsed the entire caption with just the first half rewritten and arrived at an incorrect parse (see Eg. 4.16(e)). However, once the English is corrected in the second part of the caption, the correct parse tree emerged as shown in Eg. 4.16(f).

Similar situations with other captions has led us to believe that the chances in coming up with a correct parse rest entirely on a correct English sentence and some luck in coming up with the correct parse rules and weights.

f. Prepositional Phrase Referencing

Examine the last prepositional phrase in the caption of Eg. 4.17. In reading the sentence, we would assume that "level flight" refers to the aircraft. Note that the prepositional phrase is attached to the previous noun phrase, i.e., "AIM 9." There is no way that we can associate "level flight" with aircraft syntactically. Even semantically, it is not a simple task.

Example 4.17. Prepositional referencing problem in second sentence of Caption 251707.

```
air to air view of F/A-18C BU# 163428 aircraft (nose 110 and flying eagle
on tail) with an AGM-65E laser Maverick and two Sidewinder AIM 9's (inert)
in level flight.

s(phrase(np(ng(n(noun(air to air view))),
            rnp(ofp(of(of),
                    np(ng(n(noun(F/A-18C)),
                          ng(n(noun(BU#)),
                             num(integer(163428)),
                             ng(n(noun(aircraft)),
                                app(lparen((),
                                    np(cnp(ng(n(noun(nose)),
                                              num(integer(110)))),
                                       conj(and),
                                       cnp(ng(n(noun(flying eagle))),
                                           rnp(pp(prep(on),
                                                  np(ng(n(noun(tail)))))))),
                                    rparen()))))),
                       rnp(pp(prep(with),
                              cnp(detg(det(art(an))),
                                  ng(n(noun(AGM-65E)),
                                     ng(n(noun(laser)),
                                        ng(n(noun(Maverick)))))),
                              conj(and),
                              cnp(detg(numgr(num(cardinal(two)))),
                                  ng(n(noun(Sidewinder)),
                                     ng(n(noun(AIM 9's)),
                                        app(lparen((),
                                            np(ng(n(adjp(adj(inert))))),
                                            rparen())))),
                                  rnp(pp(prep(in),
                                         np(ng(adjp(adj(level)),
                                               ng(n(noun(flight)))))))))))))))).
```

g. Dangling Preposition

The dangling preposition problem appears when we have a word that can be both an adverb and preposition and that word appears at the end of a phrase. In Eg. 4.18(a), there is no reference to associate with the term "behind." The parse tree shows that "behind" is being interpreted as an adverb. However, the adverb form has no verb with which it can be associated. The preposition rule is not used because there is no noun following the preposition. To capture the intent of "behind," we needed to manually add a noun or pronoun (e.g., "it") to the end of the sentence forcing "behind" to be a preposition. The resulting parse tree

is shown in Eg. 4.18(b). We have now explicitly established a relationship between plume and it which can be related to missile using anaphoric reference processing later.

Example 4.18. Dangling preposition problem in Caption 242099.

(a) Incorrect parse with dangling "behind."

```
missile in front of helo with large plume behind.

s(phrase(np(ng(n(noun(missile))),
            rnp(pp(prep(in front of),
                   np(ng(n(noun(helo))),
                      rnp(pp(prep(with),
                             np(ng(adjp(adj(large)),
                                   ng(n(noun(plume)))),
                                rnp(am(avp(adv(behind))))))))))))).
```

(b) Correct parse containing a reference for "behind."

```
missile in front of helo with large plume behind it.

s(phrase(np(ng(n(noun(missile))),
            rnp(pp(prep(in front of),
                   np(ng(n(noun(helo))),
                      rnp(pp(prep(with),
                             np(ng(adjp(adj(large)),
                                   ng(n(noun(plume)))),
                                rnp(pp(prep(behind),
                                       np(pronoun(it))))))))))))).
```

h. Nouns with Numbers

A problem in parsing nouns with numbers is determining whether the number associates with the preceding or the succeeding noun if one exists. In the first caption of Eg. 4.19(a), "156739" is

Example 4.19. Nouns and Numbers.

(a) Caption Examples containing Nouns with preceding and succeeding numbers.

ID	CAPTION
213856	A-7C BU# 156739 aircraft
215669	A-7B/E DVT-7 250 keas escape system test, run 2
219079	VX-5 USMC, nose 27, tail 27
237492	TAV-8B DVT-2R 225 kts seat ejection

(b) Parse tree for Caption 237492.

```
s(phrase(np(ng(n(noun(TAV-8B)),
               ng(n(noun(DVT-2R)),
                  num(integer(225)),
                  ng(n(noun(kts)),
                     ng(n(noun(seat)),
                        ng(n(noun(ejection)))))))).
```

(c) Parse tree for (b) after rewrite.

```
TAV-8B DBT-2R (225 kts) seat ejection

s(phrase(np(ng(n(noun(TAV-8B)),
               ng(n(noun(DVT-2R)),
                  app(lparen((),
                      np(detg(numgr(num(integer(225)))),
                         ng(n(noun(kts)))),
                      rparen())),
                  ng(n(noun(seat)),
                     ng(n(noun(ejection)))))).
```

associated with "BU#" and together they modify "aircraft." In the third caption, "27" can only be associate with "nose." In the two remaining captions, the question of which nouns "250" and "225" are associate with cannot be easily determined (in actuality, the correct interpretation would be "250 keas (knotsequivalent air speed) and "225 kts (knots)"). The parse rules of (11) cannot deal with this differenc because it is a semantic issue.

```
np:0.98 ==> detg, ng.                                                          (11
detg:0.85 ==> numgr.
numgr ==> num.
ng:0.98 ==> n, num, ng.
ng:0.95 ==> n, num.
```

The parse tree for Caption 237492 shown in Eg. 4.19(b) shows that 225 would be associate with the noun "DVT-2R" (a seat ejection system device number) which is the default case for such structure. The use of commas or a hyphen can be used to group the number with its correct noun. Howeve using commas has its own set of problems as we saw in Section IV.C.2.b. Adding a hyphen would force u into analyzing "225-kts" with acronyms such as "DVT-2R," "AIM-9R," etc. We decided to handle the nou and number problem as an appositive and delimit it using parenthesis. The rewritten caption and correc parse tree are shown in Eg. 4.19(c).

i. Verbals vs Verbs

A problem that appears in a number of captions concerns the parsing of verbals and actua verbs. In Eg. 4.20(a), we would interpret the word "firing" as acting as a verb. However, the parser sees i as the head noun of the noun phrase. To force the parser to treat "firing" as a verb, we must insert a "be verb before firing as shown in Eg. 4.20(b).

An alternative involving changing the weights in the parse rules so that the verb form i always preferred over the noun form can sometimes lead to erroneous results. For example, in Captio 255655, "post static firing disassembly," such a change would force "firing" to be interpreted as a ver instead of a noun.

Example 4.20. Parsing the last sentence of Caption 237492.

(a) "firing" being treated as a noun.

```
rear dummy ejecting. drogue chute above but not opened. seat rockets
firing.

s(phrase(np(ng(n(noun(seat)),
               ng(n(noun(rockets)),
                  ng(n(noun(firing)))))))).
```

(b) Forcing "firing" to be treated as a verb.

```
seat rockets are firing.

s(declarative(np(ng(n(noun(seat)),
                     ng(n(noun(rockets))))),
              vp(v(auxp(be(are)),
                   vf(prpt(prespart(firing))))))).
```

Another way to differentiate between a noun and verb that have the same root form is to use a determiner. In Eg. 4.21(a), "track" is being interpreted as a verb instead of a noun (a rocket sled rides on a track). To enable the parser to treat "track" as a noun, we manually inserted a determiner before it. When we do so, the correct parse tree results as shown in Eg. 4.21(b).

Example 4.21. Parsing the fourth phrase of Caption 217942.

(a) "track" being treated as a verb.

```
view of sled on track.

s(declarative(np(ng(n(noun(view))),
                 rnp(ofp(of(of),
                         np(ng(n(noun(sled))),
                            rnp(am(avp(adv(on)))))))),
              vp(v(pres(track))))).
```

(b) Forcing "track" to be treated as a noun.

```
view of sled on the track.

s(phrase(np(ng(n(noun(view))),
            rnp(ofp(of(of),
                    np(ng(n(noun(sled))),
                       rnp(pp(prep(on),
                              np(detg(det(art(the))),
                                 ng(n(noun(track))))))))))))).
```

Inserting a strategically placed determiner not only enables differentiating a noun from a verb, but the reverse as well. Note the underlined words in Eg. 4.22(a) which largely determine the meaning of the noun phrase. When we parse this subset, we see that "firing" appears as part of the noun phrase.

However, by inserting a determiner ("a") after "firing" in Eg. 4.22(b), we have enough of a difference to enable the parser to interpret it as a verb.

Example 4.22. Parsing Caption 29293.

(a) "firing" being treated as a noun in the underlined words in the caption.

```
air to air view of YA-4C BU# 145063 aircraft (China Lake on tail) firing
Hipeg MK 4 gun pod.

s(phrase(np(ng(n(noun(aircraft)),
              ng(n(noun(firing)),
                 ng(n(noun(gun pod)))))))).
```

(b) Forcing "firing" to be treated as a noun.

```
air to air view of YA-4C BU# 145063 aircraft (China Lake on tail) firing a
Hipeg MK 4 gun pod.

s(phrase(np(ng(n(noun(aircraft)),
         ajc(prpt(prespart(firing)),
             rvp(np(detg(det(art(a))),
                    ng(n(noun(gun pod)))))))).
```

A similar situation but one that cannot be solved without semantic help is shown in Eg. 4.23. "Shrike" is a missile and cannot fire anything, but it can be fired from an aircraft. In this parse, "firing" is taken to be a verb and not a verbal. Looking ahead into functional parsing, we will generate a verb form for "launch" whose subject is "Shrike" when in actuality we would like the noun form for "firing" with adjective "Shrike." The problem we then face is handling "firing" semantically.

Example 4.23. Incorrect interpretation of "firing" in Caption 44263.

```
Shrike firing at SCR 584 Shrike Bullpup target.

s(phrase(np(ng(n(noun(Shrike)))),
         ajc-*vtype(launch)*(prpt(prespart(firing)),
             prep(at),
             rvp-*vcase(destination)*(np(ng(n(noun(SCR 584)),
                       ng(n(noun(Shrike)),
                          ng(n(noun(Bullpup)),
                             ng(n(noun(target)))))))))).
```

Eg. 4.24(a) shows the last sentence of Caption 85488 which has another problem requiring differentiation between verbals and verbs. Before we study its parse structure, examine the parse subset of it as shown in Eg. 4.24(b). Note that "burn" appears as a noun in the prepositional phrase beginning with "of." We now slightly modify this simpler caption as shown in Eg 4.24(c). This sentence is almost identical to our original caption with the exception that "missile" is substituted for "rocket." "Burn" is now taken to be a verb based on the rules of (12). Notice that the last rule, the one that creates a noun-group

Example 4.24. Incorrect interpretation of "firing" in Caption 85488.

(a) Last sentence of caption.

```
view showing end of rocket burn.
```

(b) Parsing a subset of (a).

```
view showing end of burn.

s(phrase(np(ng(n(noun(view)))),
         ajc(prpt(prespart(showing)),
             rvp(np(ng(n(noun(end))),
                    rnp(ofp(of(of),
                            np(ng(n(noun(burn))))))))))).
```

(c) Substituting "missile" for "rocket" in (a).

```
view showing end of missile burn.

s(declarative(np(ng(n(noun(view))),
                 rnp(ajc(prpt(prespart(showing)),
                         rvp(np(ng(n(noun(end))),
                                rnp(ofp(of(of),
                                        np(ng(n(noun(missile))))))))))),
              vp(v(pres(burn))))).
```

(d) Parse tree for (a).

```
s(phrase(np(ng(n(noun(view)))),
         ajc(prpt(prespart(showing)),
             rvp(np(ng(n(noun(end))),
                    rnp(ofp(of(of),
                            np(ng(n(noun(rocket burn))))))))))).
```

phrase, carries less weight than the declarative rule, even though the declarative rule carries less weight than the phrase rule. Increasing the weights of the *ng* rule will affect not only other *ng* rules but also any phrase

```
s:0.84 ==> declarative.                                              (12)
s:0.87 ==> phrase.
declarative ==> np, vp.
phrase ==> np, ajc.
ng:0.85 ==> n, ng.
```

rules (e.g., *np*, *rnp*, etc) that reference the *ng* rules. In 95% of the cases, these weights function correctly; in this case, they are not good enough.

In order to parse the "rocket burn" phrase correctly, we need to treat it as a single lexical term in the lexicon. The parse tree that is generated is then correct. The second sentence of Caption 209362 ("Assault Breaker Martin Marietta warhead sled test") has the same problem. "Test" is interpreted as a verb instead of being the head noun of the phrase because of the dominating preference for *np*, *vp*. We can only deal with this problem by creating the lexical item "sled test."

j. Case Information for Verbs

A feature of the DBG system is the ability to add syntactic case information for verbs. Consider the first three words of Caption 234756 shown in Eg. 4.6. The verb "paint" can have in its objective case the color of the action or an actual object being painted, as in "the crew painted the drone silver." If paint is defined as in Eg. 4.25(a), then we get the parse tree in Eg. 4.25(b). Since case information can be introduced into the lexicon for a verb, we might try to handle the case of a direct object by modifying the definition of paint to that shown in Eg. 4.25(c). This additional information tells the parser that "paint" can have any general object as a direct object and filling the *object* case. Situations involving physical objects results in a *theme* case by default.

Example 4.25. Introducing Case Information to handle Caption 234756.

(a) Lexicon definition for "paint."

```
infin(paint, morph(1-a)).
```

(b) Parse tree for "drone painted silver."

```
s(phrase(np(ng(n(noun(drone)))),
          ajc(ppt(pastpart(painted)),
              rvp(np(ng(n(noun(silver)))))))).
```

(c) Lexicon definition for "paint" with case information.

```
infin(paint, morph(1-a),
  case([[dobj(gen_obj), vcase(object)],
        [dobj(phys_obj)]])).
```

(d) Parse tree for "drone painted silver" using case information.

```
s(declarative(np(ng(n(noun(drone)))),
               vp(v(past(painted)),
                  np-*object*(ng(n(noun(silver))))))).
```

Redoing the parse with this new definition gives us the parse tree in Eg. 4.25(d). However, the case information is good only for highlighting the fact that "silver" is an object because the word syntactically appears that way. In trying to arrive at a semantic representation during later analysis, we find that there is no difference between this structure and the previous parse structure.

Using case information sometimes results in a more accurate parse tree. However, cases exist in which the parse rules and weights that once worked no longer work. In Eg. 4.26(a), we parse the second sentence of Caption 219553 without the word "chip." Note that "integrated circuit" appears as the object of the sentence and the relative verb phrase commences with "on." Determining the *theme* and *location* is a straightforward task. Now examine the tree structure that results if we parse the entire caption (see Eg.

Example 4.26. Caption 219553.

```
Linda Wincn placing integrated circuit chip on solder board.
```

(a) Parsing the sentence without "chip."

```
Linda Wincn placing integrated circuit on solder board.

s(phrase(np(ng(n(noun(Linda)),
               ng(n(noun(Wincn))))),
         ajc(prpt(prespart(placing)),
             np-*vcase(theme)*(ng(n(noun(integrated circuit)))),
             rvp(am(avp(pp(prep(on),
                           np(ng(n(noun(solder board))))))))))).
```

(b) Parsing the original sentence.

```
s(phrase(np(ng(n(noun(Linda)),
               ng(n(noun(Wincn))))),
         ajc(prpt(prespart(placing)),
             np-*vcase(theme)*(ng(n(noun(integrated circuit)))),
             rvp(np(ng(n(noun(chip))),
                    rnp(pp(prep(on),
                           np(ng(n(noun(solder board))))))))))).
```

(c) Parsing the original sentence with the weights in Rules (13) and (14) switched.

```
s(phrase(np(ng(n(noun(Linda)),
               ng(n(noun(Wincn))))),
         ajc(prpt(prespart(placing)),
             np-*vcase(theme)*(ng(n(noun(integrated circuit)),
                   ng(n(noun(chip))))),
             rvp(am(avp(pp(prep(on),
                           np(ng(n(noun(solder board))))))))))).
```

4.26(b)). Notice that "chip" has been moved into the *rvp* structure. (13) and (14) are the two rules in

```
rvp:1.00 ==> np.                                          (13)

rvp:0.90 ==> am.        % adverbial modifier              (14)
```

conflict. By switching the two weights in the rules, the problem we experienced was resolved and resulted in the correct parse tree shown in Eg. 4.26(c). This tree now allows us to recognize the *theme* and *location* as easily as before. The real problem stems from the situation discussed in the previous section involving the weights for the *ng* rule. We have increased our preference for those adverbial structures beginning with a preposition over those beginning with a noun phrase.

However, we are still not out of the hole. If you examine the parse tree in Eg 4.27(a), you will see that "tank" and "target" are being separated and placed into two different syntactic structures. Our changes do no good. Yet if we parse the caption in its entirety, we get a more accurate parse tree (see Eg. 4.27(b)). If we now return to Eg. 4.26 and leave out the final prepositional phrase ("on solder board"), we notice the same exact problem. Without a final prepositional phrase and preference for an *am* rule, a *rvp* rule

Example 4.27. Caption 224159-66.

```
USAF AGM-65C laser Mavericks hitting tank targets at Eglin AFB.
```

(a) Parsing the underlined portion of the caption.

```
s(phrase(np(ng(n(noun(USAF)),
               ng(n(noun(AGM-65C)),
                  ng(n(noun(laser)),
                     ng(n(noun(Mavericks))))))),
         ajc(prpt(prespart(hitting)),
             np-*vcase(destination)*(ng(n(noun(tank)))),
             rvp(np(ng(n(noun(targets)))))))).
```

(b) Parsing the entire caption.

```
s(phrase(np(ng(n(noun(USAF)),
               ng(n(noun(AGM-65C)),
                  ng(n(noun(laser)),
                     ng(n(noun(Mavericks))))))),
         ajc(prpt(prespart(hitting)),
             np-*vcase(destination)*(ng(n(noun(tank)),
                   ng(n(noun(targets))))),
             rvp(am(avp(pp(prep(at),
                           np(ng(n(noun(Eglin AFB)))))))))))).
```

beginning with a *np* is preferred over a sequence of *ng*'s. Further experiments and adjustments to the *ng* and *rvp* weights caused more damage than good.

D. FUNCTIONAL PARSING

The output of the syntactic parser is a neutral parse tree that is fed into the functional parser. Functional parse rules are used to process different parts of the tree during functional parsing and the tree's structure largely determines the functional parse records produced. The functional parse rules are applied to the tree bottom-up, starting with the lexical items, and from left to right in the sentence. The functional parser produces records which consist of a list of syntactic roles on the left-hand side and associated values on the right-hand side. Typical roles are subject, object, preposition, adjective, etc. Some case grammar roles are also included whenever the verb case information found in the lexicon was able to be used in the syntactic parser, such as direction, source, theme, etc. The right-hand side values are either one argument predicates or words from the input sentence. Allen (1987) prescribed this type of output form as a more convenient form for creating the LF.

Eg. 4.27 shows the functional parse records for Caption 262878. A lexical item from the original caption can be substituted with the value supplied in the *fpcat* predicate defined for it in the lexicon. For example, "AIM 9C" in the original caption is replaced with "AIM-9C." Likewise, "BU#" is replaced with "bureau_no." The records can be viewed as being grouped into three categories. The first category consists of all the concepts which are represented in the type hierarchy. Next, we have all of the modifiers of those

concepts which can be adjectives, adverbs, numbers, nouns modifying nouns, etc. Finally, we have all of the relationships that the concepts might be involved in, including prepositions, appositives, and possible verb cases. This record form is easier to handle for processing nominal compounds, dealing with verb cases that are not stated explicitly, handling anaphoric resolution, etc. We have included a discussion of this phase in this report in order to establish the basis for creating the LF records and to highlight problems encountered in processing the syntactic parse tree. Notice in the listing that there is an underlying order in which certain terms appear. For example, for the noun phrase "Sidewinder AIM 9R missile," if we work our way from the bottom of the list up , we have the head noun "aircraft" appearing first, followed by the adjective noun closest to it (i.e., AIM-9R), and so on. We use this ordering and similar ones later during the logical form phase to specialize the nouns and establish whatever relationships we can.

Example 4.27. Functional Parse Records for part of Caption 262868.

```
Sidewinder AIM 9R missile mounted on F/A-18C BU# 163284 aircraft (nose
110).

PREP       =  id_of(noun(5),noun(6))
APPOS      =  app(noun(6),noun(7))
VPREP      =  on(pastpart(1),noun(6))
OBJ        =  phys_obj(pastpart(1),noun(3))
MAIN-V     =  decl(pastpart(1),assemble)
ADJS-NOUN  =  missile(noun(3),Sidewinder)
ADJS-NOUN  =  missile(noun(3),AIM-9R)
ADJS-NOUN  =  aircraft(noun(6),F/A-18C)
ADJS-NUM   =  designator(noun(7),110)
IOBJ       =  inst(noun(7),nose)
POBJ       =  inst(noun(6),aircraft)
ADJS-NUM   =  designator(noun(5),163284)
IOBJ       =  inst(noun(5),bureau_no)
GOBJ       =  inst(noun(3),missile)
```

1. Functional Parse Categories

Eg. 4.27 showed some of the basic functional parse records that we create. We now discuss these and other records in more detail.

a. General Object (GOBJ)

General Objects correspond to either a possible subject or object of a caption and are always the head noun (the last noun) of a noun phrase. A GOBJ record has the format shown in (15) where

```
GOBJ = inst(NounToken, Class)                                    (15)
```

NounToken is an instance of Class in the type hierarchy. The NounToken is a one-argument predicate that indexes the lexical item in the sentence and is taken from the parse tree tokens.

A NounToken (or VerbToken) has the general form:

PartOfSpeech(CaptionIDorQuery-SentenceNo-TokenNo)

where *PartOfSpeech* can be "noun," "pastpart," "adj," etc.; *CaptionIdorQuery* is either a numeric Caption ID or the term "query;" *SentenceNo* corresponds to the sentence number within the caption or query where the word appeared; and *TokenNo* is either the numeric index for the word within the sentence or an artificially generated token.

b. Prepositional Object (POBJ)

Prepositional Objects are those head nouns which are contained within a prepositional phrase. A POBJ record has the same right hand side structure as the GOBJ record.

```
POBJ = inst(NounToken, Class)                                   (16)
```

c. Appositive Object (AOBJ)

Appositives Objects are those head nouns that are contained in an appositive structure with the structure shown in (17).

```
AOBJ = inst(NounToken, Class)                                   (17)
```

AOBJs are treated almost identically to GOBJs. The primary motivation in distinguishing them from the GOBJs is for the handling of anaphoric reference (discussed in Section IV.F.3). We need the ability to examine a noun history list and find those general objects that are suitable as a *theme* for a verbal noun when the theme is not stated explicitly. Since we originally did not distinguish between appositive and general objects, appositive objects that appeared in the last noun phrase of a sentence were often mistaken for the *theme* of a verbal noun when examining the history list. Thus, what we actually needed was the last general object seen. In some cases, this may be the noun the appositive object refers to.

d. ID Object (IOBJ)

ID Objects are nouns followed by some alphanumeric identifier (e.g., "BU# 123456" or "nose 110"). We need to handle these nouns differently because of the need to connect the identifier that follows the noun with the noun. IOBJ records have the same format as GOBJs and POBJs.

```
IOBJ = inst(NounToken, Class)                                   (18)
```

An IOBJ record that appears in the noun phrase and functions like an adjective also has associated with it a PREP record. This record has an *id_of* predicate in its right-hand side for indicating the head noun to which the IOBJ describes. If the IOBJ should appear as an appositive instead, then an APPOS record is created.

e. Number Objects (NOBJ)

Number Objects are those words that take on a numeric form and are found in the head noun position of a noun phrase. The caption in Eg. 4.28 contains a year designator (i.e., 1989). The NOBJ record

Example 4.28. Example of a Year designator (from Caption 900304).

```
air to air view of Northrop ATB (Advanced Technology Stealth Bomber) B-2
aircraft. first public flight in late 1989.
```

structure appears the same as the previous records with one exception; the numeric string is not a class in the type hierarchy. We have defined a numeric-stamp concept in the type hierarchy that the number will be connected to temporarily until the number and its context is analyzed to see what exactly it denotes.

```
NOBJ = inst(NounToken, num_stamp)                                    (19)
```

Associated with the NounToken will be an additional attribute record which indicates the value; in the case above, the additional record will contain the value "1989."

f. Coordinates (COORD)

The Coordinates record, as the name implies, refers to longitude and latitude coordinates. An example is "4505' N x 34' E" from Caption 180657. We presently treat the coordinates as one unit. The COORD record structure appears very similar to the NOBJ record. However, instead of a numeric-stamp concept, we use the *coordinate* class. The actual values will then become an attribute of the *coordinate* class.

```
COORD = inst(NounToken, coordinates)                                 (20)
```

g. Dates (DATE)

The Dates record appears exactly like the coordinates record except that the *date* concept is used instead of the *coordinates* concept as shown in (21). Unlike the coordinates record, we do some

```
DATE = inst(NounToken, date)                                         (21)
```

preprocessing of the date and put it into format (22) to allow it to be stored and manipulated like a date data type in some of the commercial relational databases (e.g., Ingres and Oracle).

```
day-month-year                                                       (22)
```

h. General Acts (GACT)

General Acts refer to the verbals. We treat them separately because we need to discover what the verbal is acting upon or refers to (i.e., the *theme* of the act). For example, in the phrase "view of missile," "missile" would be the *theme* of the "view." GACT's are defined in the lexicon by having their functional parse predicate set to *event* (e.g., *fp(event)*). The GACT record has the same format as the other noun objects.

```
GACT = inst(NounToken, Class)                                        (23)
```

i. Main Verbs (MAIN-V)

The Main Verb record, as the name suggests, identifies the main verb or verbs of a sentence. A MAIN-V record has the format shown in (24). The Mood value can be one of *decl* (declarative), *imp*

```
MAIN-V = Mood(VerbToken, Class)                                      (24)
```

(imperative), or *ques* (question). All captions and queries have taken the declarative form. The VerbToken and Class arguments work similarly to what we have seen for nouns.

j. Noun Adjectives (ADJS-NOUN)

The Noun Adjectives are those nouns that precede the head noun of a noun phrase and act like adjectives. We saw in the Eg. 4.28 that both "Sidewinder" and "AIM 9R" are Noun Adjectives for the head noun "aircraft." ADJS-NOUN records have the format shown in (25). The Fp value is taken from the *fp*

```
ADJS-NOUN = Fp(NounToken, NounString)                    (25)
```

predicate defined in the lexicon. The NounToken refers to the head noun which the NounString modifies. These records are used to either further specialize the head noun based on the type hierarchy ordering or infer correlations between the head noun and the Noun Adjective. The NounString can itself be a class when it appears as the head noun in a noun phrase.

k. Number Adjectives (ADJS-NUM)

A Number Adjective is an alphanumeric identifier that follows the (IOBJ) noun. In the earlier example, the numeric values in the strings "BU# 163284" and "nose 110" are Number Adjectives. Caption 85486 ("ship target #E") has an alphabetic character as the adjective. ADJS-NUM records have the format of (26). The NounToken refers to the IOBJ noun token that immediately precedes the identifier; hence, number adjectives function just like adjectives to the noun. The NumIdString is alphanumeric string value of the identifier.

```
ADJS-NUM = designator(NounToken, NumIdString)            (26)
```

l. Quantifier Adjectives (ADJS-QUANT)

Quantifier Adjectives are either numbers or word quantifiers that modify a head noun; they actually precede the noun. Examples are "closeup" in "closeup side view," "3/4" in "3/4 front overall view," and "24" in "24 inert bombs." ADJS-QUANT records have the format shown in (27) where the NounToken refers to the head noun of the noun phrase and NumIdString is some number or word quantifier.

```
ADJS-QUANT = quant(NounToken, NumIdString)               (27)
```

m. Adjectives (ADJS)

The Adjectives records are those words that are defined in the lexicon as adjectives. The record format is shown in (28). The Fp value is taken from the *fp* predicate defined in the lexicon. The NounToken refers to the head noun which the AdjString modifies.

```
ADJS = Fp(NounToken, AdjString)                          (28)
```

n. Adverbs (ADV)

The Adverb record (29) contains those words that are defined in the lexicon as adverbs. Their record format is similar to Adjectives.

```
ADV = Fp(VerbToken, AdvString)                           (29)
```

o. Prepositions (PREP)

The Prepositions record identifies those head nouns that are connected by a preposition. For example, in the phrase "the soldering assembly area in Michelson Lab" from Caption 219558, the head nouns "area" and "Michelson Lab" are connected by the preposition "in." The PREP record has the format:

```
PREP = Preposition(NounToken1, NounToken2)                    (30)
```

The Preposition predicate is either the lexical item or the *fpcat* predicate if defined. Some of the prepositions have their *fp* value defined as *rev* (reverse) for situations where we wish to state a relation just one way instead of two. This is discussed further in the next section.

p. Number (NUM)

The NUM record is used to indicate that a particular noun is plural. The record format is shown in (31) where NounToken is the plural head noun.

```
NUM = plural(NounToken)                                       (31)
```

q. Implicit Verb Case Information

The lexicon and syntactic parse rules allow the ability to explicitly encode syntactic case information within the lexicon. The parse rules contain specific rules as to how they will be used. Some parse rules, however, are written generically and do not require case information be supplied. We must, though, examine the resulting parse tree structure more carefully in order to infer case information.

We have presently defined four types of implicit cases: SUBJ, OBJ, VPREP, and VTIME. The Subject record indicates possible syntactic subject(s) for the verb specified in a MAIN-V record. The SUBJ record has the format:

```
SUBJ = Fp(VerbToken, NounToken)                               (32)
```

The Fp value is taken from the *fp* predicate defined in the lexicon. The VerbToken relates back to the MAIN-V record and the NounToken refers to the head noun of the subject.

The Object record indicates possible syntactic object(s) for the verb specified in the MAIN-V record. The OBJ record is identical to the SUBJ record.

```
OBJ = Fp(VerbToken, NounToken)                                (33)
```

The Verb Preposition record is used to designate those prepositions and head nouns that could possibly fill in a case slot for a verb. The record structure is a mixture of the PREP and SUBJ/OBJ records.

```
VPREP = Preposition(VerbToken, NounToken)                     (34)
```

The Verb Time record is associated with verbal nouns and is used to indicate a time frame for when the verbal noun occurred with respect to the verb specified in the MAIN-V record. Examples include "view of aircraft after hit" from Caption 228795 and "side view of plane during banking over NWC ranges" from Caption 232745. The VTIME record has the format:

```
VTIME = TimePrep(NounToken, VerbToken)                        (35)
```

r. *Explicit Verb Case Information*

Explicit case information can be supplied for certain structures following the verb. For these situations where the parse rules use the case information defined in the lexicon, the following record types are generated: DIR (direction), GOAL, OBJECT, SOURCE, and THEME. The record structures are similar to VPREP.

2. Complications

Functional parsing for our application requires using the lexical tokens (word place identifiers in a sentence) to designate variables in slot-assertion notation; these tokens were not used in the original DBG functional parse output. These tokens provide a convenient method of connecting concepts horizontally in the type hierarchy for indicating a concept's roles (cases) and correlations in a caption. In the original DBG system, templates (frames) were predefined for each of the major concepts that were to be captured. Each functional parse record was then processed to determine where it would fit in a predefined template. Values were related to one another only when they were placed within the appropriate template. As we indicated earlier, we have chosen not to use predefined templates at this time.

Most of the parsing problems are determining the correct neutral parse tree. The functional parse rules rely heavily on a correct tree. To a large extent, these rules are very straightforward. However, problems have arisen and the following sections discuss these problems.

a. *Sequence of Prepositional Phrases*

We define this problem as the determination of prepositional modifiers in a sentence containing a sequence of prepositional phrases. Examine the prepositions in the caption of Eg. 4.29. Based on the prepositions, we must derive the relationships between each of the prepositional objects and their corresponding preceding nouns. A plausible interpretation is shown in Eg. 4.29(a). Arriving at this solution mechanically is not obvious and sometimes requires a detailed knowledge of the various objects in the sentence and their interrelationships. Taking a more syntactic approach to the problem, it appears that a more general solution is possible. Assume a preposition connects its object noun not with the immediately preceding noun, but the last noun preceding the first preposition; we associate the preposition with the subject directly. This assumption would result in the clauses in Eg. 4.29(b).

Example 4.29. Interpreting the prepositions in Caption 10851.

```
Bullpup missile on bomb skid in hangar bay outside of elevator aboard the
USS Lexington CVA-16.
```

(a) Possible interpretation of prepositions

```
on(missile, 'bomb skid')
in('bomb skid', 'hangar bay')
outside('bomb skid', elevator)
on('bomb skid', CVA-16)
```

(b) Syntactic handling of prepositions

```
on(missile, 'bomb skid')
in(missile, 'hangar bay')
outside(missile, elevator)
on(missile, CVA-16)
```

(c) Replacing "outside of elevator" with "with F/A-18's"

```
on(missile, 'bomb skid')
in('missile', 'hangar bay')
with('hangar bay', 'F/A-18')
on('missile', CVA-16)
```

This general rule has been implemented and works a majority of the time, but runs into problems when dealing with the prepositions "with" and "of." For these two cases, the nouns preceding and following the preposition must be joined. For example, suppose we replaced "outside of elevator" with "with F/A-18's." The interpretation in Eg. 4.29(c) would then seem reasonable.

An additional problem with "with" is determining its influence on adjacent prepositions. In Eg. 4.30(a), the prepositional phrase following the "with" phrase ("from inside tower") refers to the source of the "view" whereas in Eg. 4.30(b), "on tail" refers to "SH." Distinguishing between the references for the two prepositions is impossible during syntactic parsing. Adding a rule that checks to see if a prepositional phrase containing an "of" precedes the "with" is of no help as can be seen in the modification shown in Eg. 4.30(c).

Example 4.30. Captions involving "with."

(a) Second sentence of Caption 124.

`view of runway with aircraft from inside tower.`

(b) Third sentence of Caption 227282.

`F-4J BU# 158378 aircraft with SH on tail in background.`

(c) Modification of (b).

`view of F-4J BU# 158378 aircraft with SH on tail in background.`

We surmise that there are no simple syntactic solutions for handling "with" and similarly related words, even if we add rules to handle the lexical item "with" embedded by generic nouns. We need to examine "with" in the context of the sentence to know how to handle it correctly; i.e., we need to know the specific categories of nouns surrounding "with" and what correlations may exist between them. Hence, we can conclude that additional information (i.e., semantic information) needs to be included during the

syntactic parsing phase, implying context-sensitivity. This capability can be handled in a future rewrite of the syntactic parser to include semantic information.

Eg. 4.31(a) provides a somewhat different perspective of the sequence problem. Examine the treatment of the underlined nominal compound. "G-1 Range" is a specific range at the NAWS China Lake, and "T-5" is a tower at the "G-1 Range." We can create a correlation in the type hierarchy between these two classes that indicates that "T-5" is a *part_of* the "G-1 Range" as well as a complement correlation *has_part*. The predicate "tower(noun(5), T-5)" shows "T-5" acting as an adjective-noun to "G-1 Range." Predicate "over(noun(3), noun(5))" has "ejection" occurring over the "G-1 Range" as opposed to over the "T-5" tower. Not exactly the correct meaning, but we could perform noun-phrase analysis during creation of the logical form to make explicit the fact that a relationship exists between "T-5" and "G-1 Range" to show that "T-5" is somehow involved in the view.

Example 4.31. Preposition Problem involving Nominal Compounds (from Caption 85486).

(a) Original Caption.

```
RAPEC seat ejection over T-5 G-1 Range from QF-9F drone aircraft.

PREP        =  over(noun(3),noun(5))
PREP        =  from(noun(3),noun(8))
ADJS-NOUN   =  project(noun(3),RAPEC)
ADJS-NOUN   =  phys_obj(noun(3),seat)
ADJS-NOUN   =  tower(noun(5),T-5)
ADJS-NOUN   =  aircraft(noun(8),QF-9F)
ADJS-NOUN   =  phys_obj(noun(8),drone)
POBJ        =  inst(noun(8),aircraft)
POBJ        =  inst(noun(5),G-1 Range)
GACT        =  inst(noun(3),ejection)
```

(b) Splitting the Nominal Compound into two prepositional phrases.

```
RAPEC seat ejection over T-5 tower at G-1 Range from QF-9F drone aircraft.

PREP        =  over(noun(3),noun(5))
PREP        =  at(noun(3),noun(6))
PREP        =  from(noun(3),noun(9))
ADJS-NOUN   =  project(noun(3),RAPEC)
ADJS-NOUN   =  phys_obj(noun(3),seat)
ADJS-NOUN   =  tower(noun(5),T-5)
ADJS-NOUN   =  aircraft(noun(9),QF-9F)
ADJS-NOUN   =  phys_obj(noun(9),drone)
POBJ        =  inst(noun(9),aircraft)
POBJ        =  inst(noun(6),G-1 Range)
POBJ        =  inst(noun(5),tower)
GACT        =  inst(noun(3),ejection)
```

If we were now to change the caption as in Eg. 4.31(b), we get a slightly different interpretation. We now have predicates "at(noun(3), noun(6))" and "over(noun(3), noun(5))" showing

"ejection" is occurring at the "G-1 Range" and over the "T-5 tower." We have gained a somewhat more accurate semantics by introducing a prepositional phrase into the sequence, but have lost the means to discover that "T-5" is related to the "G-1 Range" during noun phrase analysis. This is also different than the previous interpretation which had ejection occurring over the G-1 Range. We could recover this information if we compared each successive pair of head nouns in the prepositional phrases to each other, looking at the prepositions, and seeing if certain correlations can be inferred between the two that correspond to the preposition. Having accomplished this, we then must deal with correcting "ejection" at "G-1 Range."

b. Conjunctive Nouns

There are two problems when dealing with conjunctive noun phrases. The first problem is the handling of plurals. Eg. 4.32(a) shows the parse tree for a noun phrase involving a plural head noun. What the caption implicitly states is that the term "missiles" encompasses both "Bat" and "Standard Arm"

Example 4.32. Plural Conjunctive Nouns (from Caption 110169).

`Bat and Standard Arm missiles with A-6A aircraft.`

(a) Parst tree for underlined noun phrase.

```
s(phrase(np(cnp(n(noun(Bat))),
            conj(and),
            cnp(ng(n(noun(Standard Arm)),
                   ng(n(noun(missiles)))))))).
```

(b) Functional parse records for underlined noun phrase.

```
ADJS-NOUN  =  missile(noun(3),Standard Arm)
GOBJ       =  inst(noun(3),missile)
NUM        =  plural(noun(3))
GOBJ       =  inst(noun(1),Bat)
```

(c) Parse tree for entire caption.

```
s(phrase(np(cnp(n(noun(Bat))),
            conj(and),
            cnp(ng(n(noun(Standard Arm)),
                   ng(n(noun(missiles)))),
                rnp(pp(prep(with),
                       np(ng(n(noun(A-6A)),
                             ng(n(noun(aircraft))))))))))).
```

(d) Functional parse records for entire caption.

```
PREP       =  with(noun(3),noun(5))
ADJS-NOUN  =  missile(noun(3),Standard Arm)
ADJS-NOUN  =  aircraft(noun(5),A-6A)
POBJ       =  inst(noun(5),aircraft)
GOBJ       =  inst(noun(3),missile)
NUM        =  plural(noun(3))
GOBJ       =  inst(noun(1),Bat)
```

yet the parse tree shows the plural "missiles" being associated with "Standard Arm" only. If we look at the functional parse (see Eg. 4.32(b)), we find that this is indeed the case as only "Standard Arm missile" is taken to be plural. Do we drop the "NUM" record to indicate this? Suppose there is one Bat missile but several Standard Arm missiles with the aircraft? We do not have a simple solution to this problem.

The second problem with the conjunctive nouns is handling the preposition that follows the last noun in the conjunction. We have already seen this type problem during syntactic parsing in Section C. Let's now examine the parse tree for the entire caption (see Eg. 4.32(c)). This example shows one of the first problems with handling "with," which is the subject of the next section also. Do we interpret the caption to say that the "Standard Arm" missile(s) are with the aircraft or that both missiles are with the aircraft? If we had used a conjunction ("and") instead of the "with" we would have a less ambiguous sentence for parsing ("Bat missile, Standard Arm missile, and A-6A aircraft."). Eg. 4.32(d) shows the functional parser records produced for the original caption.

By examining the PREP record and looking ahead to creation of the logical form, we know that "with" can be used to designate a *part_of/has_part* relation as in a "missile" can be *part_of* an "aircraft" (specifically an attack aircraft, A-6A). However, we are asserting that only the Standard Arm missile is part of the aircraft because of Standard Arm modifying the GOBJ missile. Applying the preposition to both involves elevating the functional parse rules to examining for *cnp* nodes as well, checking for a plural form in the last *ng*, then checking to see if the head nouns for each of *cnp* noun phrases all have the same class (for this case only). This is more of a workaround than fixing the problem with the parse tree. Hence, we have additional evidence for moving this semantic type checking forward and using it during syntactic parsing to derive a better parse tree to avoid the problem in the functional parse.

Similar type problems appear in Eg. 4.33(a). Examples of sentences in which the structure and functional parse records produced are correct are shown in Eg. 4.33(b).

Example 4.33. Additional Conjunctive Noun Examples

(a) Captions with similar conjunctive noun problems.

<u>ID</u>	<u>CAPTION</u>
69812	... NASA employee and Richard Fulmer with the batteries and ...
168579	closeup view of Walleye II bomb and data link pod <u>on</u> A-7E aircraft.
213798	excellent view of hangar and control tower <u>in</u> background.

(b) Captions where conjunctive nouns are handled correctly.

<u>ID</u>	<u>CAPTION</u>
230834	closeup view of aircraft and center of sea site.
238225	... Skywarrior aircraft with Standard Missile II-N and EX-62 TDD in nose
241452	... aircraft (nose 622 and Marines on tail)

c. Sentences/Phrases connected using "with"

A more complicated problem involving "with" appears when it acts as a conjunction of two sentences/phrases as shown in Eg. 4.34(a). Eg. 4.34(b) shows the functional parse records. The predicate "with(noun(2),noun(4))" has "wing" and "Bat missile" being connected syntactically. Looking ahead into creation of the logical form, this would result in a *part_of* relationship being established between the two

Example 4.34. Sentences connected using "with."

(a) Caption 110169.

```
Standard Arm mounted on wing with Bat missile sitting on ground.
```

(b) Functional Parse records for (a).

```
VPREP       =  on(pastpart(1),noun(2))
PREP        =  with(noun(2),noun(4))
VPREP       =  on(prespart(1),noun(5))
OBJ         =  missile(pastpart(1),noun(1))
SUBJ        =  phys_obj(prespart(1),noun(4))
MAIN-V      =  decl(prespart(1),inhabit)
MAIN-V      =  decl(pastpart(1),assemble)
ADJS-NOUN   =  missile(noun(4),Bat)
POBJ        =  inst(noun(5),land)
POBJ        =  inst(noun(4),missile)
POBJ        =  inst(noun(2),wing)
GOBJ        =  inst(noun(1),Standard Arm)
```

(c) Functional Parse records for (a) with "with" replaced by "and."

```
VPREP       =  on(pastpart(1),noun(4))
VPREP       =  on(pastpart(1),noun(2))
VPREP       =  on(prespart(1),noun(5))
OBJ         =  missile(pastpart(1),noun(1))
SUBJ        =  phys_obj(prespart(1),noun(4))
MAIN-V      =  decl(prespart(1),inhabit)
MAIN-V      =  decl(pastpart(1),assemble)
ADJS-NOUN   =  missile(noun(4),Bat)
POBJ        =  inst(noun(5),land)
POBJ        =  inst(noun(4),missile)
POBJ        =  inst(noun(2),wing)
GOBJ        =  inst(noun(1),Standard Arm)
```

(d) Examples of captions with similar problems.

ID	CAPTION
209862	excellent view over desert terrain with bank of clouds in background.
213798	helicopter lifting off with excellent view of hangar and control tower in background.
242099	missile in front of helo with large plume behind missile.

objects since a wing can have as one of its parts a missile. This interpretation would be incorrect. Substituting the conjunction "and" for "with" leads to a different interpretation (see Eg. 4.34(c)). The

predicate "on(pastpart(1), noun(4))" incorrectly indicates that the Standard Arm missile is mounted on the Bat missile in addition to the wing. The only viable solution at this point is to replace the "with" with a period and create two separate sentences. The interpretations then come out correct for both sentences.

Examples of similar problems are shown in Eg. 4.34(d).

With almost every problem we discover and believe we have solved, there are always exceptions. Eg. 4.35 shows one of those cases. This sentence is clearly syntactically similar to the previous caption with respect to the use of "with." However, in this case, the building actually has an antenna tower

Example 4.35. Proper handling of "with" (from Caption 213528).

```
front view of building with antenna tower completed.

PREP        =  of(noun(1),noun(2))
PREP        =  with(noun(2),noun(4))
OBJ         =  phys_obj(pastpart(1),noun(4))
MAIN-V      =  decl(pastpart(1),complete)
ADJS-NOUN   =  phys_obj(noun(4),antenna)
POBJ        =  inst(noun(4),tower)
GOBJ        =  inst(noun(2),building)
GACT        =  inst(noun(1),front view)
```

as part of it. Looking ahead to creation of the logical form, it is correct to infer a *part_of* relation between the building and the tower.

Another problem involving "with" involves any prepositions that may follow it. In Eg. 4.36(a), note that the second *PREP* record has "AIM-9M" being located on the "wingtip." During creation of the logical form, we could then easily map this to a *location* record.

Example 4.36. Prepositions following "with."

(a) Preposition related records from third sentence of Caption 255577.

```
F/A-18A BU# 161713 aircraft (nose 101) with Sidewinder AIM-9M on right
wingtip.

PREP  =  with(noun(3),noun(6))
PREP  =  on(noun(6),noun(7))
POBJ  =  inst(noun(7),wingtip)
POBJ  =  inst(noun(6),AIM-9M)
GOBJ  =  inst(noun(3),aircraft)
```

(b) Preposition related records from second sentence of Caption 258795.

```
Cdr Antonio with night vision goggles in cockpit.

PREP   =  with(noun(2),noun(4))
PREP   =  in(noun(4),noun(5))
POBJ   =  inst(noun(5),cockpit)
POBJ   =  inst(noun(4),goggles)
GOBJ   =  inst(noun(2),Antonio)
```

Now let's introduce a different sentence with the same syntactic structure as we have just seen (see Eg. 4.36(b)). In this case though we have that the "goggles" are located in the "cockpit" instead of "Antonio." Handling this type of situation requires we look at the "with" more closely and attempt to find any relationships between its two arguments that may apply to the following preposition. Note, that we have already examined other problems with "with" in section (a). By examining the role of "with," we see that aircraft has the Sidewinder missile mounted on it, while Antonio has the goggles mounted on him. In the first caption, wingtip is a part of the aircraft so we may infer that it should apply to the preceding noun phrase. However, cockpit is not a part of Antonio so we should infer that it apply not to the immediate preceding noun phrase, but the one before it. We have not implemented these semantic rules in the functional parser as they present a major shift in its processing flow; by incorporating such rules, we could have actually generated a more accurate parse tree had we been able to get alternatives from the syntactic parser.

d. Adjectives as Verbs

The caption of Eg. 4.37 shows a situation where we have an adjective acting as a verb during parsing due to parse rule (36). To handle this possibility, we had to rewrite the lexical definition for "operational" (and other verbal adjectives) to refer back to a verb concept.

ajc:0.90 ==> adjp, am. (36)

The lexicon definition of "operational" is now defined as:

```
adj(operational, fp(event), fpcat(perform)).
```

The verb "operate" has its semantic category set to "perform" as well. Using this definition, we get the functional parse records shown in the example. The MAIN-V record represents the fact that "operational" refers to a verb which reduces to the "perform" act. Because we are now working with a verb structure, we can use the verb-related rules to find the two subjects, "missile(adj(1), noun(1))" and "organization(adj(1), noun(3))," and the location, "in(adj(1), noun(4))" where the event is occurring.

Example 4.37. Handling Adjectives as Verbs (from Caption 10862).

```
Sparrow III (Guardian of the Skies) operational with Seventh Fleet in
western Pacific.

APPOS      =  app(noun(1),noun(2))
VPREP      =  in(adj(1),noun(4))
SUBJ       =  missile(adj(1),noun(1))
SUBJ       =  organization(adj(1),noun(3))
MAIN-V     =  decl(adj(1),perform)
ADJS       =  position(noun(4),western)
POBJ       =  inst(noun(4),Pacific Ocean)
POBJ       =  inst(noun(3),Seventh Fleet)
GOBJ       =  inst(noun(2),Sparrow 3)
GOBJ       =  inst(noun(1),Sparrow 3)
```

Other adjectives being treated similarly include "attached," "camouflaged," and "complete."

e. Prepositional Inverses

In Section IV.D.1.o, we mentioned setting the *fp* predicate to *rev* for those situations where we wish to store inverse relations just one way. The purpose of doing this is to simplify the handling of relations when matching a query against the caption. The handling of prepositional inverse relationships is handled within the lexicon and the functional parser. For the caption in Eg. 4.38, the preposition "above" is inverse to "below." Rather than have both "above" and "below" existing in the final logical form, we do some simplification. "Below" and "above" are defined in the lexicon as:

```
prep(above, fp(rev), fpcat(under)).

prep(below, fpcat(under)).
```

because of their similarity to the word "under." When processing the parse tree, the functional parser recognizes that a reversal of the arguments is called for by the *fp* predicate with a semantic substitution defined by the *fpcat* predicate. The functional parser then produces the output shown in Eg. 4.38(a).

If we had not defined "above" this way, the functional parser would have produced "above(noun(2),noun(3))." Now, however, the "above" predicate is replaced by "under" and the arguments

Example 4.38. Handling Prepositional Inverses (from Caption 234756).

almost complete <u>view</u> <u>of</u> <u>Armitage</u> <u>Field</u> <u>above</u> <u>aircraft</u>.

(a) Functional Parse records of underline words in caption.

```
PREP   =  of(noun(1),noun(2))
PREP   =  under(noun(3),noun(2))
POBJ   =  inst(noun(3),aircraft)
GOBJ   =  inst(noun(2),Armitage Field)
GACT   =  inst(noun(1),view)
```

(b) Parse tree for the entire caption.

```
s(imperative(am(avp(adv(almost))),
             ivp(imph(infin(complete)),
                 rvp(np(ng(n(noun(view))),
                        rnp(ofp(of(of),
                                np(ng(n(noun(Armitage Field))),
                                   rnp(pp(prep(above),
                                          np(ng(n(noun(aircraft)))))))))))))).
```

(c) Parse tree for the entire caption without the leading "almost."

```
s(phrase(np(detg(det(quant(complete))),
            ng(n(noun(view))),
            rnp(ofp(of(of),
                    np(ng(n(noun(Armitage Field)))
                       rnp(pp(prep(above),
                              np(ng(n(noun(aircraft))))))))))))).
```

(d) Functional Parse records (c).

```
PREP          =  of(noun(1),noun(2))
ADJS-QUANT    =  quant(noun(1),full)
GOBJ          =  inst(noun(2),Armitage Field)
GACT          =  inst(noun(1),view)
```

are switched. This approach works for PREP records but not VPREP records because VPREP records contain a verb as the first argument. The prepositional inverses involve the second argument of the predicate record and the syntactic object of the verb. These substitutions must be done during creation of the logical form. Note a parsing problem with this sentence - the handling of the word "complete." It is defined as:

```
adj(complete, fp(event)).
infin(complete, morph(3-a)).
quant(complete, fpcat(full)).
```

and the parse tree generated is shown in Eg. 4.38(b). Had the caption been "complete view of Armitage Field above aircraft," the syntactic parse structure would have been that of Eg. 4.38(c) with the functional parse records shown in Eg. 4.38(d). However, since "almost" is defined as an adverb, it signals the existence of an imperative structure resulting in the erroneous parse structure as seen above.

E. THE TYPE HIERARCHY

Creating the logical form from the functional parse records entails mapping the individual functional parse records to class/subclass concepts arranged in a type hierarchy using inheritance. A particular real-world entity of a class is termed an *instance* [Charniak 1987]. This hierarchy allows us to specialize class instances through examination of the adjective nouns, establish correlations between classes (e.g., *part_of*, *owned_by*, *employee_of*, etc.), and defining rules for case relations (e.g., *agent*, *source*, *theme*, etc.). The overall appearance then becomes one of a semantic network. Brachman (1983a) goes into a lengthy discussion on the issue of what constitutes an IS-A inheritance link. Without getting into all of the issues, our usage is consistent with that described in Rowe (88).

Slots defined at higher classes in the hierarchy are inherited by subclasses. If a slot is defined at both the class and subclass and we are working with an instance of a subclass, then we will first use the slots at the subclass. If none apply, we in turn look up the hierarchy to find the inherited values. In the remainder of this section, we discuss the design and structure of the type hierarchy. A listing is contained in Appendix E.

1. Design Philosophy

The type hierarchy contains both noun and verb concepts. We have used the NASA Thesaurus (1988), Defense Technical Information Center Thesaurus (1990), and the NAWCWPNS Authority List as references in defining the noun concepts. The hope being to reference a common thesaurus across existing and future systems. Classes are based on *logically proper names* and not *definite descriptions* [Frixione 89]. Logically proper names refer to the same object in any possible world (e.g., "George Washington,"

"Gettysburg," etc.) versus definite descriptions which can be associated with different objects in different possible worlds (e.g., "the smallest horse," the biggest Redwood tree," etc.). Some of the verb concepts have similarities with the CD ACTs described earlier [Schank 75] with the verb hierarchy reflecting works from Allen (1987) and Charniak and McDermott (1987).

A major decision in designing the structure of the type hierarchy involves making the distinction between what is a class (object) and what is an instance of the class. Figure 4.1 shows a structure in which the level of generalization or specialization has been artificially established by distinguishing between proper names and physical concepts. In Figure 4.1, "Sidewinder AIM-9C" is treated as an instance of the class "air-to-air missile." However, "Sidewinder AIM-9D" would also be treated as an instance of the same class, a situation that ignores characteristics of the "AIM-9" that distinguish one version from another. These characteristics would also be objects (classes) and to indicate that one class has some relationship to another class just because one of its subclasses does may be erroneous with respect to the characteristics. For example, two different versions of the AIM-9x missiles may have used two different types of seekers.

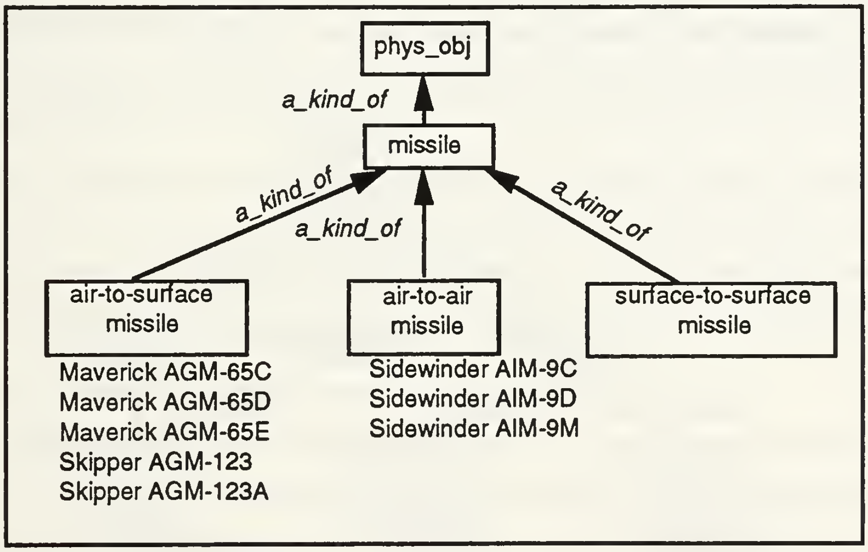

Figure 4.1. Type Hierarchy Design I - Certain concepts being treated as Class Instances.

Let's briefly examine an indexing issue which will be covered in more detail in the next chapter. Consider a query that contains "AIM-9M," a kind of Sidewinder missile. Any search process should be able to find immediately those images that contain the "M" specific version of the missile because of its mere presence in the query. Likewise, if "AIM-9" is supplied in the query, the search process should find all "AIM-9" images. But it will not find the "AIM-9M" or other versions unless a wildcard (e.g., "*" as in UNIX) is used. We would like the system in the future to perform this search automatically. Likewise, if

the query contained "Sidewinder," then we would like the system to find not only all the captions that contained the word "Sidewinder," but those that contained some variant of "AIM-9x" or some other designator for "Sidewinder" if one was used.

A further complexity of the Figure 4.1 approach is the classifying of token instances (e.g., noun(1)) for the xOBJ and MAIN-V records created by the functional parser. To keep the specific class that appears in those records, it would be necessary to either create the tokens as instances of the instances shown in Figure 4.1 (e.g., "Sidewinder AIM-9M" would be an instance of the class "air-to-air missile" and noun(1) would have to be an instance of "Sidewinder AIM-9M") or associate tokens with the instance using slot values. Both of these approaches introduce a degree of artificiality.

Figure 4.2 shows an alternative design where we treat each proper name and object concept as a separate class. As a result, the number of classes in the system is drastically increased. However, the advantages appear to outweigh the disadvantages. As we will see in the next chapter, both designs maintain the same total number of index records, although the number of index files will differ. For cases where a class may have multiple interpretations (e.g., "Sidewinder" designates a program office as well as a type of missile), a second parent is introduced and an ordering scheme is specified to handle multiple inheritance.

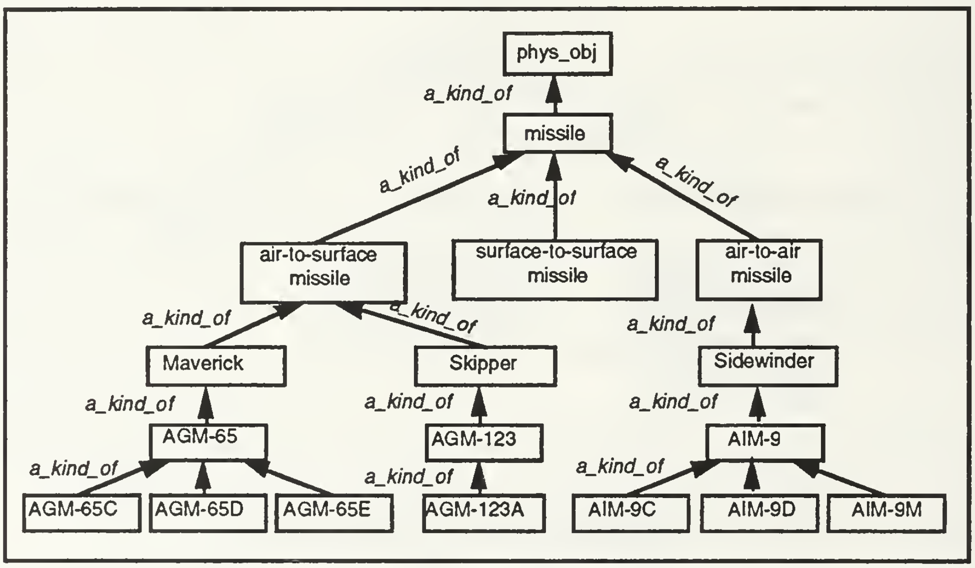

Figure 4.2. Type Hierarchy Design II - All concepts treated as classes.

2. Structure

The type hierarchy class structure is specified using *a_kind_of* predicates of the form:

```
a_kind_of(Child, ChildSense, Parent, ParentSense, Links).
```

where:

```
Child          child class
```

ChildSense	sense identifier to distinguish between multiple sense (definitional) records of the same child; field is formed of a Part of Speech Identifier (either 'noun' or 'infin') and a numeric number.
Parent	parent class
ParentSense	similar to ChildSense
Links	list of [Class, Sense] records to represent semantic link relationships between members of the same class or between noun and verb forms.

We have defined five inherited slots to create the logical form and use as references in matching: *inner_cases*, *transforms*, *infer*, *correlations*, and *case_vals*. The *inner_cases* slot lists the inner cases (Allen, 1987) a verb or a noun may take (e.g., *agent*, *location*, *time*, etc.). The *transforms* slot contains rule information to allow a token involved in one specific case to be transformed (translated) to a different case. The *infer* slot also contains rule information to allow additional cases to be generated based on the existence or nonexistence of specific existing cases. The *correlations* slot lists how two classes are related to one another; example relations are *part_of* and *has_part*, *owned_by* and *owner_of*, etc. The *case_vals* slot is used to record the actual inner_case values and correlations values pertaining to the caption found by processing the functional parse records.

Methods are defined primarily to generate the logical form, create the index records, perform the matching, set the case_vals, as well as a number of auxiliary functions involving the five slots. Slots and methods are associated with the root class (*e_r_concept*) as well as various subclasses when more specialized processing is needed. All slots and methods are subject to inheritance based on the class/subclass ordering.

F. CREATING THE LOGICAL FORM

As we indicated earlier, the functional parse records are mapped to the type hierarchy to create the logical form. We have chosen not to use predefined templates (frames) at this time to analyze how effective IR can be done without them as these structures must also be manually created and require more insight and work than just the lexicon and type hierarchy. Thus semantic knowledge as to what type of functional parse record goes where is not a factor. We seek an approach to make the system more general in handling a wide variety of captions. In processing the functional parse records, a specific order must be followed. First, the noun (*x*OBJ, COORD, DATE) and verb (MAIN-V) records are processed. These records are used to create the class instances based on the token variables contained within the records since all other records reference an instance in one way or another. Next, the modifiers (adjectives and adverbs) are processed to enable us to specialize a class instance to a particular subclass instance. Lastly, the relationship records (prepositions and case specific records) are processed given that now we have the most specialized class instances to work with.

We originally used one processing pass to handle all of the functional parse records. However, processing of the noun phrases revealed that at least two passes were needed. The adjective nouns and appositives in some cases were not actually specializing the head noun but indicating relationships, as in "US Navy aircraft" where "US Navy" is the *owner_of* the "aircraft" and "aircraft wing" where "wing" is a *part_of* the "aircraft." Another situation requiring a second pass involved specializing adjective nouns that preceded the head noun, as in "SCO-96 MK 82 bomb electric heat cookoff test 454692" where "MK 82" is a kind of "bomb." Hence, the first pass handled those nouns that specialized the head nouns as well as other preceding adjective nouns and the second pass handled the remaining adjectives, nouns, and relationships. Eg. 4.39 shows the LF records after the first two passes for Caption 262868. Later when we cover anaphoric resolution, we will discuss the need for a third pass. Appendix F contains the logical form records for all of the captions used in this research.

Example 4.39. Logical Form Records for part of Caption 262868 (see Eg. 4.27).

```
inst(noun(262868-1-3),AIM-9R).
attribute(noun(262868-1-6),has_part(noun(262868-1-7))).
attribute(noun(262868-1-6),inst(noun(262868-1-5))).
inst(noun(262868-1-6),F/A-18C).
attribute(noun(262868-1-5),163284).
inst(noun(262868-1-5),bureau_no).
theme(pastpart(262868-1-1),obj(noun(262868-1-3))).
location(pastpart(262868-1-1),on(noun(262868-1-6))).
activity(pastpart(262868-1-1),assemble).
attribute(noun(262868-1-7),110).
inst(noun(262868-1-7),nose).
```

1. Logical Form Structure

The logical form records are based on case grammar representations and are all two argument predicate structures. We have used case grammar constructs as described by Haas (1990). The record formats are shown in Table 4.1. The first argument of all the records indicates the instance being described. The second argument is either a modifier (e.g., strings or number) filling the case predicate, an instance that is related to the first through a correlation, an instance that is again related to the first but through some type of syntactic role or preposition, or a class for indicating what the instance token is an instance of.

Most of the instance identifiers themselves are based upon the lexical identifier from the parse tree (e.g., noun(2), pastpart(1), etc). The identifiers are modified to include the caption number and a sentence number. This modification allows the type hierarchy to hold multiple sentences as well as multiple captions and the query.

The remainder of this chapter discusses the more unique processing capabilities for handling captions. We will forgo discussing the more straightforward and simpler processing.

TABLE 4.1. LOGICAL FORM RECORD FORMATS.

accompaniment(Instance1,inst(Instance2))	Instance1 is accompanied by Instance2
activity(Instance,Class)	Instance is a verb of type Class
agent(Instance1,obj(Instance2))	An agent for Instance1 is Instance2
attribute(Instance,Qualifier)	An attribute of Instance is specified by Qualifier
attribute(Instance1,inst(Instance2))	An attribute of Instance1 is Instance2
attribute(Instance1,Corr(Instance2))	An attribute of Instance1 is Instance2 as specified by the correlation Corr
destination(Instance1,Corr(Instance2))	A destination for Instance1 is Instance2 as specified by the correlation Corr
direction(Instance,Value)	A direction for Instance is specified by Value
force(Instance1,obj(Instance2))	A force for Instance1 is Instance2
goal(Instance1,Corr(Instance2))	A goal for Instance1 is Instance2 as specified by the correlation Corr
inst(Instance,Class)	Instance is a noun of type Class
instrument(Instance1,obj(Instance2))	An instrument for Instance1 is Instance2
location(Instance1,Corr(Instance2))	A location for Instance1 is Instance2 as specified by the correlation Corr
manner(Instance1,Corr(Instance2))	A manner in which Instance1 was done is Instance2 as specified by correlation Corr
object(Instance1,obj(Instance2))	An object for Instance1 is Instance2
quantity(Instance,Value)	A quantity for Instance is specified by Value
quantity(Instance,Measure(Value))	A quantity for Instance is specified by Value of type Measure
reason(Instance1,obj(Instance2))	A reason for Instance1 is Instance2
source(Instance1,Corr(Instance2))	A source for Instance1 is Instance2 as specified by the correlation Corr
state(Instance,Value)	A state for Instance is specified by Value
theme(Instance1,obj(Instance2))	A theme for Instance1 is Instance2
time(Instance,Value)	A time for Instance is specified by Value
time(Instance1,Corr(Instance2))	A time for Instance1 is Instance2 as specified by the correlation Corr

2. Noun Phrase Analysis

As we mentioned earlier, the captions we have examined have very rich noun phrase structures. The handling of nominal compounds within a noun phrase is a difficult task. We have attempted to analyze the phrases looking primarily for specialization and correlation information. However, our analysis is not as in-depth as the one suggested by Gay and Croft (1990) for nominal compounds. As one shortcut, our approach treats certain sequences of nouns as a single index term (e.g., "U S Navy," "aircraft carrier," "Beavertail cactus," etc.). This methodology reflects the current Photo Lab database administrator's view of how index terms are handled and what images should be retrieved given certain index terms. Braun (1976) described some early work on automatically creating similar index phrases from natural language text.

In determining what should be treated as one phrase as opposed to multiple phrases, we have had to look at how these sequences interact with other nouns. These sequences in many cases have involved more than two nouns in a sequence. Rules and strategies for handling these situations are similar to ideas

presented in Finin (1986). His notion of concepts and subconcepts is the same as our class type hierarchy structure. Finin also expressed the need to look at the context in which a nominal compound is used in order to better understand the relationships of the noun constituents. In our case of captions, looking at previous sentences has served to resolve references to specializations and generalizations of a noun found in the nominal compound.

The following sections discuss and show examples of how the noun phrases and nominal compounds are handled for the captions we have seen.

a. Specialization

In almost all of the captions encountered, we have seen cases where subclass information for a particular head noun is supplied. In Eg. 4.40(a), the underlined noun phrase contains three nouns representing three classes. The hierarchy specifies that "missile" is the highest class, followed by "Sidewinder" which is a subclass of "missile" and "AIM 9R" which is a subclass of "Sidewinder." In actuality, "AIM 9R" is a subclass of "AIM 9" which in turn is a subclass of "Sidewinder." The functional parse records for the underlined nouns are shown in Eg. 4.40(b).

Example 4.40. Specialization of Head Noun within a Noun Phrase.

(a) Caption 262865

```
3/4 front view of Sidewinder AIM 9R missile on stand.
```

(b) Functional Parse records for (a).

```
ADJS-NOUN    =  missile(noun(3),Sidewinder)
ADJS-NOUN    =  missile(noun(3),AIM-9R)
GOBJ         =  inst(noun(3),missile)
```

(c) Logical Form records derived from (b).

```
inst(noun(262865-1-3), 'AIM-9R')
```

In the process of mapping these three records to the type hierarchy, the "GOBJ" expression is processed first because we need to establish a class instance in the type hierarchy. Processing of "AIM-9R" (corrected spelling from the *fpcat* predicate) then reveals it as a subclass of "missile" and results in the "noun(3)" term being specialized to "AIM-9R." Finally the "Sidewinder" term is processed, but since "AIM-9R" is a subclass of "Sidewinder," no further work is necessary. The resulting logical form record produced is shown in Eg. 4.40(c).

Specialization not only applies to the adjective nouns modifying the head noun, but to an adjective noun modifying another adjective noun in a noun phrase as well. Examine the underlined nouns Eg. 4.41(a). The head noun in the noun phrase is the nominal compound "cookoff test" ("cookoff test" is a kind of "test") (see Eg. 4.41(b)). We can conclude that "bomb" indicates the object (theme) of the test, and

Example 4.41. Specialization of an Adjective-Noun within a Noun Phrase.

(a) Second sentence of Caption 178012.

```
SCO-96 MK 81 bomb electric heat cookoff test 454692.
```

(b) Functional Parse records for (a).

```
ADJS-NOUN              =  test(noun(5),SCO-96)
ADJS-NOUN              =  bomb(noun(5),MK-81)
ADJS-NOUN              =  phys_obj(noun(5),bomb)
ADJS                   =  char(noun(5),electric)
ADJS-NOUN              =  abst_obj(noun(5),heat)
ADJS-NUM               =  designator(noun(5),454692)
IOBJ                   =  inst(noun(5),cookoff test)
```

(c) Logical Form records derived from (b).

```
inst(noun(178012-1-a3),MK-81).
attribute(noun(178012-1-5),454692).
attribute(noun(178012-1-5),inst(noun(178012-1-a2))).
attribute(noun(178012-1-5),electric).
theme(noun(178012-1-5),obj(noun(178012-1-a3))).
attribute(noun(178012-1-5),SCO-96).
inst(noun(178012-1-5),cookoff test).
inst(noun(178012-1-a2),heat).
```

that "MK- 81" is a specific type of bomb. We thus have an embedded case of specialization. In this particular case, since both "MK-81" and "bomb" do not specialize the test, we would have a new noun instance created for "bomb" (since it is processed before "MK-81"). The following "MK-81" record would then specialize the bomb instance to "MK-81" in the type hierarchy. We are then left with discovering the relationship between "MK-81" and "cookoff test" (which we examine in a later section). The rationale for creating an instance for "MK-81" is to allow the caption to be indexed by other specialized nouns (like "MK-81") as opposed to just the head noun ("cookoff test"). The logical form records produced are shown in Eg. 4.41(c).

Hence, for both cases we see that the instance corresponding to the most specialized noun of a nominal compound related via class/subclass relationship is the one that is actually recorded in the type hierarchy. There is also a bug in this representation dealing with the nominal compound "electric heat."

b. Correlations

In addition to specialization, analysis of the nominal compound can reveal correlations between the nouns. In Eg. 4.42(a), the noun phrase "FBM guided missile submarine SSG-N1" involves five terms - four nouns and an adjective. The NASA Thesaurus defines "guided missile submarine" as an index term and a subclass of "submarines." This allows us to reduce the number of terms we have to analyze from five to three. As discussed in the previous section, "SSG-N1" would be classified as a subclass of "guided missile submarine" which would be specialized to simply "SSG-N1" leaving us with identifying

the relationship between "FBM" (Fleet Ballistic Missile) and "SSG-N1." The functional parse records are shown in Eg. 4.42(b).

Example 4.42. Finding correlations within a Noun Phrase.

(a) Second sentence of Caption 10880.

```
artist conception of FBM guided missile submarine SSG-N1 (nuclear program).
```

(b) Functional Parse records for underlined nouns in (a).

```
ADJS-NOUN   =  missile(noun(5),FBM)
ADJS-NOUN   =  ship(noun(5),guided missile submarine)
GOBJ        =  inst(noun(5),SSG-N1)
```

(c) Logical Form records derived from (b).

```
attribute(noun(10880-2-5),has_part(noun(10880-2-a1))).
inst(noun(10880-2-5),SSG-N1).
inst(noun(10880-2-a1),FBM).
```

Within the type hierarchy, we have defined a correlation slot to identify how classes correlate with one another. For our example above, we have the following slot definitions for the classes "FBM" and "guided missile submarine." As can be seen, establishing correlations between the previous two terms

```
slot('FBM',noun-1,correlations,
      [c(part_of,'guided missile submarine')]).

slot('guided missile submarine', noun-1, correlations,
      [c(has_part, 'FBM')]).
```

involves "SSG-N1" inheriting this fact from "guided missile submarine." Through this discovery, we are able to assert a *part_of/has_part* attribute between the two nouns.

From the functional parse records, we would first create an instance of the class "FBM" by deriving a new token. This new instance would then be used to establish the correlation between the "SSG-N1" instance. The resulting logical form is shown in Eg. 4.42(c).

Other examples include "aircraft wing," "J52 fuel ingestion" (where J52 is the type of engine), and "RAPEC seat ejection" (involves two correlations: RAPEC and ejection, seat and ejection).

c. Inferring Ownership

In addition to the correlation slots, there exist mapping rules from the functional parse records (in particular ADJS-NOUN) that examine the arguments to see what in general is true about them. We have added an explicit rule that when an organization noun precedes some physical object noun in the functional parse record listing, infer an *owned_by* relationship. Examine the caption in Eg. 4.43(a) and functional parse records in Eg. 4.43(b). We know through specialization and the handling of appositives that the noun phrase reduces to "Northrop B-2." The mapping rules would then examine the records

Example 4.43. Finding ownership relations within a Noun Phrase.

(a) Noun phrase from Caption 900304.

```
Northrop ATB (Advanced Technology Stealth Bomber) B-2 aircraft.
```

(b) Functional Parse records for (a).

```
APPOS        =   app(noun(5),noun(3))
ADJS-NOUN    =   organization(noun(5),Northrop)
ADJS-NOUN    =   aircraft(noun(5),ATB)
ADJS-NOUN    =   aircraft(noun(5),B-2)
GOBJ         =   inst(noun(5),aircraft)
GOBJ         =   inst(noun(3),ATB)
```

(c) Logical Form records derived from (b).

```
attribute(noun(900304-1-3),owned_by(noun(900304-1-a1))).
inst(noun(900304-1-3),B-2).
inst(noun(900304-1-a1),Northrop).
```

"organization(noun(5), Northrop)" and the specialization to "B-2" and arrive at an ownership property. The resulting logical form records are shown in Eg. 4.43(c).

Other examples include "VC-3 aircraft" (where VC-3 is the squadron), "USN UH-2A," and "Martin Marietta Assault Breaker."

A slightly different problem is when there are two organizational nouns functioning as adjectives. In Eg. 4.44(a) and Eg. 4.44(b), both "USMC" and "VMA-513" (squadron) modify "AV-8A" (which is specialized from aircraft). For certain organization combinations we have been able to identify correlations between them. In this case, we know "VMA-513" is *part_of* the the "USMC" and have stated this explicitly in the *correlations* slot in the type hierarchy. The logical forms records produced are shown in Eg. 4.44(c).

Example 4.44. Two consecutive organizational nouns within a Noun Phrase.

(a) Noun phrase from Caption 163030.

```
FAE weapons on USMC VMA-513 AV-8A BU# 158389 Harrier aircraft.
```

(b) Functional parse records involving organization and head noun.

```
ADJS-NOUN    =   organization(noun(8),USMC)
ADJS-NOUN    =   organization(noun(8),VMA-513)
ADJS-NOUN    =   aircraft(noun(8),AV-8A)
ADJS-NOUN    =   aircraft(noun(8),Harrier)
POBJ         =   inst(noun(8),aircraft)
```

(c) Noun phrase from Caption 163030.

```
attribute(noun(163030-1-a22),part_of(noun(163030-1-a23))).
inst(noun(163030-1-a22),VMA-513).
attribute(noun(163030-1-8),owned_by(noun(163030-1-a22))).
inst(noun(163030-1-8),AV-8A).
inst(noun(163030-1-a23),USMC).
```

d. Inferring Themes

In some captions, we will have a verbal as the head noun of the noun phrase. In Eg. 4.45(a), "ejection" is the head noun for "RAPEC" and "seat." From the functional parse records for these terms (see Eg. 4.45(b)), we have a general rule that if a physical object (seat) is modifying an event, then infer that the physical object is the theme of the event. However, we must first create an instance for the physical object and then infer the theme. The resulting logical form is shown in Eg. 4.45(c).

Example 4.45. Verbal as the Head Noun of a Noun Phrase.

(a) Noun phrase from Caption 85486.

```
RAPEC seat ejection over T-5 G-1 Range from QF-9F drone aircraft.
```

(b) Functional Parse records for underlined nouns in (a).

```
ADJS-NOUN    =  phys_obj(noun(3),seat)
GACT         =  inst(noun(3),ejection)
```

(c) Logical Form records derived from (b).

```
theme(noun(85486-2-3),of(noun(85486-2-a2))).
inst(noun(85486-2-3),ejection).
inst(noun(85486-2-a2),seat).
```

Other examples include "SCO-96 MK 81 bomb electric heat cookoff test 454692," "seat firing," "parachute deployment," and "escape system test."

Eg. 4.46(a) shows a situation where we have a verbal acting as an appositive to the previous head noun. The functional parse for the appositive case is shown in Eg. 4.46(b). "Round 4" is treated as a verbal, similar to "view." If the caption had read "round 4 of AV-8A Harrier (600 keas) escape system test," then we would have inferred the *theme* of "round 4" to be "test." We likewise do the same for appositive verbals and infer the *theme* to be the head noun the verbal is an appositive to. This results in the logical form shown in Eg. 4.46(c).

Example 4.46. Verbal as Appositive to the Head Noun of a Noun Phrase.

(a) Noun phrase from Caption 180657.

```
AV-8A Harrier (600 keas) escape system test (round 4).
```

(b) Functional Parse records for Appositive and Head Noun.

```
APPOS       =  app(noun(5),noun(6))
ADJS-NUM    =  designator(noun(6),4)
IOBJ        =  inst(noun(6),round)
GACT        =  inst(noun(5),test)
```

(c) Logical Form records derived from (b).

```
attribute(noun(180657-1-6),4).
theme(noun(180657-1-6),of(noun(180657-1-5))).
inst(noun(180657-1-6),round).
inst(noun(180657-1-5),test).
```

e. *Inferring Agents*

Similar to the situation with themes, we may have a human being modifying an event as shown in Eg. 4.47(a). From the functional parse output for these terms (see Eg. 4.47(b)), we also have a general rule such that if a human being (artist) is modifying an event, then infer that the human being is the agent for the event. Again, we must first create an instance for the human being and then infer the agent case. The resulting logical form appears in Eg. 4.47(c).

Example 4.47. Inferring an Agent in a Noun Phrase.

(a) Second sentence of Caption 10880.

```
artist conception of FBM guided missile submarine SSG-N1.
```

(b) Functional Parse records for underlined nouns in (a).

```
ADJS-NOUN    =  human being(noun(2),artist)
GACT         =  inst(noun(2),conception)
```

(c) Logical Form records derived from (b).

```
inst(noun(10880-2-a4),artist).
agent(noun(10880-2-2),by(noun(10880-2-a4))).
inst(noun(10880-2-2),conception).
```

f. *Inferring Isa's*

Isa's are inferred when we are dealing with two objects that are loosely related. Examine the underlined noun phrase in Eg. 4.48(a). Using the type hierarchy, "aircraft" specializes to "QF-9F." "Drone," however, is not a part of the aircraft hierarchy. Anything that can remotely be controlled can be considered a drone. Hence, we must arrive at a relationship between the two objects. We have as one of the default

Example 4.48. Inferring Isa Relationships in a Noun Phrase.

(a) Second sentence of Caption 85486.

```
RAPEC seat ejection over T-5 G-1 Range from QF-9F drone aircraft.
```

(b) Functional Parse records for underlined nouns in (a).

```
ADJS-NOUN    =  aircraft(noun(8),QF-9F)
ADJS-NOUN    =  phys_obj(noun(8),drone)
POBJ         =  inst(noun(8),aircraft)
```

(c) Logical Form records derived from (b).

```
attribute(noun(85486-2-8),isa(noun(85486-2-a5))).
inst(noun(85486-2-8),QF-9F).
inst(noun(85486-2-a5),drone).
```

general mapping rules that if two nouns are physical objects and we cannot infer any correlation or other relationship, establish an *isa* relationship between the two. For the functional parse structure in Eg. 4.48(b), we arrive a the logical form records shown in Eg. 4.48(c).

Other similar examples include "tank target," "fire plume," "cockpit station," and "target (pole with checkerboard)."

g. *Inferring States*

Verb forms acting as adjectives in noun phrases are inferred as *states*. In the noun phrase Eg. 4.49(a), the word "mod" ("modified") is actually a verb but it is acting as an adjective to "A-7." It would not make sense to infer a verb phrase with subject and object. Rather we make the assumption that "mod" says something about the state of the "A-7," specifically that it has been modified. In the lexicon, we define "mod" as an adjective with *fp* "event" and *fpcat* "modify" to indicate it is based on a verb form along with its semantic representation. From the functional parse records (see Eg. 4.49(b)), we produce the following logical form records of Eg. 4.49(c).

Example 4.49. Inferring States in a Noun Phrase.

(a) First noun phrase of Caption 181709.

FLIR <u>mod</u> A-7.

(b) Functional Parse records for (a).

```
ADJS-NOUN       =  instrument(noun(2),FLIR detector)
ADJS            =  event(noun(2),modify)
GOBJ            =  inst(noun(2),A-7)
```

(c) Logical Form records derived from (b).

```
state(noun(181709-1-2),modify).
attribute(noun(181709-1-2),has_part(noun(181709-1-a38))).
inst(noun(181709-1-2),A-7).
inst(noun(181709-1-a38),FLIR detector).
```

Other examples include "camouflaged paint," "the gutted aircraft," "unfilled chute," and "the receding fireball."

h. Default Case

The default for nouns that do not fit any of the previous categories is to become a modifier to the noun. Modifiers can be either descriptive terms or instances. Examine the treatment of the appositive "nuclear program" in Eg. 4.50. Note that the value of the *attribute* is a descriptive word.

Example 4.50. Modifiers in an Appositive.

(a) Appositive from Caption 10880.

```
SSG-N1 (nuclear program).
```

(b) Functional Parse records for (a).

```
ADJS     =  char(noun(6),nuclear)
GOBJ     =  inst(noun(6),program)
```

(c) Logical Form records derived from (b).

```
attribute(noun(10880-2-6),nuclear).
inst(noun(10880-2-6),program).
```

In Eg. 4.51, "BU#" is created as an instance of the class "bureau_no," 149033 becomes an attribute of the instance of "bureau_no," and the instance of "bureau_no" becomes an attribute of "UH-2A" after specialization.

Example 4.51. Numeric Modifiers in nominal compound.

(a) Numeric Modifier from Caption 161044.

```
USN UH-2A BU# 149033 BQM control helicopter
```

(b) Functional Parse records for underlined nouns in (a).

```
PREP        =  id_of(noun(3),noun(5))
ADJS-NOUN   =  aircraft(noun(5),UH-2A)
GOBJ        =  inst(noun(5),helicopter)
ADJS-NUM    =  designator(noun(3),149033)
IOBJ        =  inst(noun(3),bureau_no)
```

(c) Logical Form records derived from (b).

```
attribute('noun(161044-1-5)',inst('noun(161044-1-3)')).
inst('noun(161044-1-5)','UH-2A').
attribute('noun(161044-1-3)','149033').
inst('noun(161044-1-3)',bureau_no).
```

Numbers that precede a noun fill the *quantity* case. In Eg. 4.52, "250 keas" is a measurement, similar to "55 mph." "Keas" (Knot Equivalent Air Speed) indicates a performance metric of 250 for "test."

Example 4.52. Numeric Modifiers in nominal compound.

(a) Numeric Modifier from Caption 215669.

```
A-7B/E DVT-7 (250 keas) escape system test (run 2).
```

(b) Functional Parse records for underlined nouns in (a).

```
APPOS         =  app(noun(5),noun(3))
ADJS-QUANT    =  quant(noun(3),250)
GACT          =  inst(noun(5),test)
AOBJ          =  inst(noun(3),keas)
```

(c) Logical Form records derived from (b).

```
quantity(noun(q-2-5),keas(250)).
inst(noun(q-2-5),test).
```

Other noun phrases that behave similarly are "pod at 17 degrees tilt (down)," "seat ejection (225 kts) at 1900'N x 21'W," and "120 mm projectile at 3740 feet per second."

i. ***Coordinates***

Coordinates in a caption are handled similarly to nouns. As we saw in Caption 180657 (shown in Eg. 4.53(a)), the coordinate is treated as a single lexical unit. In the functional parse records for the underlined words (see Eg. 4.53(b)), we have to associate the coordinates with "firing" because of the PREP record involving the "at." Ordinarily when we have an "inst" predicate, we would create the token (first argument) as an instance of the second argument (class). In this case, however, we cannot do this because we have no class with this name. Instead, we create the token as an instance of the class "coordinate" and then make the coordinate values an attribute of the token. The handling of the PREP record then is straightforward. The logical form records produced are shown in Eg. 4.53(c).

Example 4.53. Handling Coordinates.

(a) Numeric Modifier from Caption 215669.

```
synchro firing at 4505'N x 34'E from camera 44.
```

(b) Functional Parse records for underlined nouns in (a).

```
PREP   =  at(noun(1),coordinate(1))
ADJS   =  char(noun(1),synchronous)
COORD  =  inst(coordinate(1),4505 ' N x 34 ' E)
GACT   =  inst(noun(1),firing)
```

(c) Logical Form records derived from (b).

```
attribute(coordinate(180657-1-1),at(4505 ' N x 34 ' E)).
inst(coordinate(180657-1-1),coordinate).
attribute(noun(180657-1-1),synchronous).
location(noun(180657-1-1),at(coordinate(180657-1-1))).
inst(noun(180657-1-1),firing).
```

(d) Possible alternative Logical Form records.

```
attribute(noun(180657-1-1),synchronous).
location(noun(180657-1-1),coordinate(4505 ' N x 34 ' E)).
inst(noun(180657-1-1),firing).
```

One could argue that it would make more sense to eliminate the coordinate instance and just make the coordinate the value of the *location* as in Eg. 4.53(d). The resulting records would be a lot cleaner. The problem is two-fold. The first problem is knowing when to get rid of the coordinate instance in the logical form. The second problem is based on matching. Suppose a query used the term "coordinate." If we removed the coordinate reference from the logical form, then we would not have any key for knowing which captions contained coordinates and which did not. We would have to revert to a sequential search of all of the captions.

j. Dates

Dates are handled similarly as coordinates. Eg. 4.54 shows the records created for Caption 257055.

Example 4.54. Handling Dates.

(a) Date from Caption 257055.

```
intercept with QF-86 drone on April 12, 1989.
```

(b) Functional Parse records for underlined nouns in (a).

```
PREP       =  on(noun(3),date(1))
ADJS-NOUN  =  aircraft(noun(3),QF-86)
DATE       =  inst(date(1),12-apr-1989)
POBJ       =  inst(noun(3),drone)
```

(c) Logical Form records derived from (b).

```
time(noun(257055-3-3),on(date(257055-3-1))).
inst(noun(257055-3-3),QF-86).
attribute(date(257055-3-1),12-apr-1989).
inst(date(257055-3-1),date).
```

3. Anaphoric Reference

Recall that our goal in representing instances is always to use the most specialized class in the type hierarchy whenever possible. Consider the word "missiles" in the second sentence of the caption in Eg. 4.55(a). As part of the mapping process when encountering any new noun, we attempt to discover if a more specialized class for it was previously used. In the case of Caption 10862 we know that "Sparrow 3" ("Sparrow III" is mapped to "Sparrow 3" by the *fpcat* predicate) is a subclass of "missile" in our type hierarchy. As a result, we specialize the instance of "missile" to an instance of "Sparrow 3" and record this

substitution for future reference. This bookkeeping allows all references to the "missile" instance in the second sentence to be resolved to the "Sparrow 3" instance.

Eg 4.55(b) shows a subset of functional parse records which list the major records associated with "missiles" and "Sparrow 3." The first sentence has tokens for both "Sparrow 3" and its appositive (its semantic representation or *fpcat* is "Sparrow 3" as well). The second sentence uses the more general term "missiles." We make the assumption that "missiles" refers to "Sparrow 3" in the previous sentence and change all references for "missiles" to "Sparrow 3" in the second sentence. Eg. 4.55(c) shows the logical form.

Example 4.55. Anaphoric Reference processing involving two head nouns.

(a) Anaphoric reference between "Sparrow III" and "missiles" in Caption 10862.

```
Sparrow III (Guardian of the Skies) operational with Seventh Fleet in
western Pacific. four missiles on underside of F3H-1 BU# 137010 aircraft
(Point Mugu 7010 on tail). air to air view from side during firing of one
missile.
```

(b) Functional Parse records influencing/influenced by the underlined nouns in (a).

```
          Sentence 1

APPOS       =  app(noun(1),noun(2))
SUBJ        =  missile(adj(1),noun(1))
MAIN-V      =  decl(adj(1),perform)
GOBJ        =  inst(noun(2),Sparrow 3)
GOBJ        =  inst(noun(1),Sparrow 3)

          Sentence 2

PREP        =  under(noun(1),noun(4))
ADJS        =  num(noun(1),4)
ADJS-NOUN   =  aircraft(noun(4),F-3H-1)
POBJ        =  inst(noun(4),aircraft)
GOBJ        =  inst(noun(1),missile)
NUM         =  plural(noun(1))
```

(c) Logical Form records derived from (b).

```
inst(noun(query-2-4),F-3H-1).
quantity(noun(query-1-1),plural(4)).
location(noun(query-1-1),under(noun(query-2-4))).
inst(noun(query-1-1),Sparrow 3).
agent(adj(query-1-1),subj(noun(query-1-1))).
activity(adj(query-1-1),perform).
```

The previous discussion focused on anaphoric reference involving two nouns in two separate sentences. We must also handle the situation of a noun in the first sentence and a related adjective noun in the second. Examine the "aircraft" references in Eg 4.56(a) and the functional parse records in Eg. 4.56(b). In the first sentence, the aircraft reference specializes to "C-130A." In analyzing the noun phrase of "aircraft

wing," we discover that a *part_of/has_part* correlation exists between the wing and aircraft. As we discussed in Section IV.F.C.2, we would attempt to create an instance of "aircraft" to correlate with the instance of "wing." However, before doing this, we should see if a more specific aircraft reference exists and use it instead of just "aircraft." In doing this, we find the "C-130A" instance and thus establish the correlation between it and wing. The logical form for "aircraft wing" is shown in Eg. 4.56(c).

Example 4.56. Anaphoric Reference processing involving head noun and an adjective-noun.

(a) Anaphoric reference between underlined nouns in Caption 161045.

```
BQM drone mounted on DC-130A BU# 158228 VC-3 aircraft. closeup view of
drone and aircraft wing.
```

(b) Functional Parse records for the underlined nouns in (a).

```
          Sentence 1

ADJS-NOUN    =  aircraft(noun(6),C-130A)
POBJ         =  inst(noun(6),aircraft)

          Sentence 2

ADJS-NOUN    =  phys_obj(noun(4),aircraft)
GOBJ         =  inst(noun(4),wing)
```

(c) Logical Form records derived from (b).

```
inst(noun(161045-1-6),C-130A).
attribute(noun(161045-2-4),part_of(noun(161045-1-6))).
inst(noun(161045-2-4),wing).
```

A more complicated situation appears in Eg. 4.57(a) involving "aircraft" in the first sentence and specific types of aircraft in the second sentence. The problem is mapping the predicates that reference the instance of aircraft in the first sentence and handling a plural number of aircraft. Eg. 4.57(b) shows the functional parse records for both sentences.

The first problem involves applying the "VX-5" adjective and the fact that they are on the "Sinkex operation" to all three aircraft. Adjusting this information for the first aircraft, "A-6E," is trivial as it means we now have a more specialized class of aircraft than before. Thus, we can just substitute the instance of "aircraft" for a new instance of "A-6E." The problem now, however, is that we have lost the

Example 4.57. Anaphoric Reference processing involving a general noun followed by specific noun references.

(a) Anaphoric reference between underlined nouns in Caption 209862.

```
air to air view of VX-5 aircraft on Sinkex operation. A-6E, A-7E's, and A-
4M carrying LGB's (Laser Guided Bombs).
```

(b) Functional Parse records for (a).

```
          Sentence 1

PREP         =  of(noun(1),noun(3))
PREP         =  on(noun(3),noun(5))
ADJS-NOUN    =  organization(noun(3),VX-5)
ADJS-NOUN    =  operation(noun(5),Sinkex)
GACT         =  inst(noun(5),operation)
GOBJ         =  inst(noun(3),aircraft)
GACT         =  inst(noun(1),air-to-air view)

          Sentence 2

APPOS        =  app(noun(4),noun(5))
SUBJ         =  aircraft(prespart(1),noun(3))
SUBJ         =  aircraft(prespart(1),noun(2))
SUBJ         =  aircraft(prespart(1),noun(1))
THEME        =  obj(prespart(1),noun(4))
MAIN-V       =  decl(prespart(1),carry)
AOBJ         =  inst(noun(5),LGB)
NUM          =  plural(noun(5))
GOBJ         =  inst(noun(4),LGB)
NUM          =  plural(noun(4))
GOBJ         =  inst(noun(3),A-4M)
GOBJ         =  inst(noun(2),A-7E)
NUM          =  plural(noun(2))
GOBJ         =  inst(noun(1),A-6E)
```

(c) Logical Form records derived from (b).

```
attribute(noun(209862-2-3),owned_by(noun(209862-1-a7)).
location(noun(209862-2-3),on(noun(209862-1-5)).
inst(noun(209862-2-3),A-4M).
quantity(noun(209862-2-4),plural).
inst(noun(209862-2-4),LGB).
quantity(noun(209862-2-2),plural).
attribute(noun(209862-2-2),owned_by(noun(209862-1-a7)).
location(noun(209862-2-2),on(noun(209862-1-5)).
inst(noun(209862-2-2),A-7E).
attribute(noun(209862-2-1),owned_by(noun(209862-1-a7)).
location(noun(209862-2-1),on(noun(209862-1-5)).
inst(noun(209862-2-1),A-6E).
inst(noun(209862-1-a7),VX-5).
theme(noun(209862-1-5),obj(noun(209862-2-3)).
theme(noun(209862-1-5),obj(noun(209862-2-2)).
theme(noun(209862-1-5),obj(noun(209862-2-1)).
inst(noun(209862-1-5),Sinkex).
theme(noun(209862-1-1),obj(noun(209862-2-3)).
theme(noun(209862-1-1),obj(noun(209862-2-2)).
theme(noun(209862-1-1),obj(noun(209862-2-1)).
inst(noun(209862-1-1),air-to-air view).
agent(prespart(209862-2-1),obj(noun(209862-2-1)).
agent(prespart(209862-2-1),obj(noun(209862-2-2)).
agent(prespart(209862-2-1),obj(noun(209862-2-3)).
theme(prespart(209862-2-1),obj(noun(209862-2-4)).
activity(prespart(209862-2-1),carry).
```

"aircraft" instance to allow the same processing to apply to "A-7E" and "A-4M." Hence, we need to retain the instance of "aircraft" and decide when it is no longer needed.

The second problem is applying "air to air view" to all three aircraft to arrive at the *themes*. The determination of theme for the view is handled solely while processing the first sentence. Again, resetting the instance of aircraft to A-6E will retain the theme information for the A-6E. However, we have no simple way of mapping the theme to the other aircraft short of examining all slot values for all instances in the type hierarchy to see if the instance to aircraft was referenced. If it does, then we can duplicate the slot value assignments for each of the remaining aircraft. Again, this requires that we retain the aircraft instance until all the aircraft subclasses have been processed.

Determining when to keep and when to get rid of a more generalized instance can be done at the time the instances are created. We can scan the remaining tokens to be created and see if there are further subclasses of the same generalized class. If so, we retain the generalized class until all the subclasses have been processed. We also set flags to indicate the subclass instances that need to be treated equally. This special processing entailing a sort of universal quantification requires a third pass to handle the logical form. This pass then handled the similarity processing between subclasses and removed the generalized subclass when it was no longer needed.

The logical form records that result from this specialized processing as well as the basic processing are shown in Eg. 4.57(c).

Eg. 4.58 shows the reverse situation. In this case, we have multiple specific objects being mentioned first followed by a more generalized plural class identifier (i.e., "missiles"). We need "full view" to apply to more than one missile. In this case though, we can recognize the plural form of "missiles" and attempt to apply "full view" to all of the missile specialization instances. However, this simple solution would result in an erroneous interpretation if the sentence "Sidewinder AIM 9M no longer in use" appeared before the last sentence of the caption; we would be applying "full view" to a missile that may not be in the image. We leave this as an unresolved issue. (Note: presently, the anaphoric resolution rules associate "missile" with the a specific type instance based on the order the instance was created; in this case - "AIM-9R").

Example 4.58. Anaphoric Reference processing involving specific nouns followed by a general noun reference.

```
Sidewinder AIM 9R and Sidewinder AIM 9L mounted on F/A-18A BU# 162396
aircraft (nose 105 and tail 5 with flying eagle). ... full view of missiles
and rear of aircraft.
```

As we have seen thus far, anaphoric reference can be tricky. Eg. 4.59(a) shows a different twist to the problem. When processing station 7 and station 8, we recognize the fact that "station" appeared in a

previous sentence. As a result, we would throw out the second and third instances of station and keep only one, with 6, 7, and 8 modifying it using *attribute* records as the logical form records in Eg. 4.59(b) show.

Example 4.59. Resolving a series of identical nouns followed by unique identifiers.

(a) Anaphoric reference between underlined nouns in Caption 253960.

```
program Skyray. air to air view of TA-7C BU# 156738 aircraft (nose 700).
view below aircraft. HS camera at station 6. Walleye BTV (Ballistic Test
Vehicle) at station 7. Skyray instrumentation pod at station 8.
```

(b) Erroneous Logical Form records for underlined nouns in (a).

```
attribute(noun(253960-1-2),8).
attribute(noun(253960-1-2),7).
attribute(noun(253960-1-2),6).
inst(noun(253960-1-2),station).
```

(c) Functional Parse records for phrases containing underlined nouns in (a).

```
              Sentence 4

PREP        =  at(noun(1),noun(2))
ADJS-NUM    =  designator(noun(2),6)
IOBJ        =  inst(noun(2),station)
GOBJ        =  inst(noun(1),HS camera)

              Sentence 5

PREP        =  at(noun(2),noun(4))
ADJS-NUM    =  designator(noun(4),7)
IOBJ        =  inst(noun(4),station)
GOBJ        =  inst(noun(2),BTV)

              Sentence 6

PREP        =  at(noun(2),noun(3))
ADJS-NUM    =  designator(noun(3),8)
IOBJ        =  inst(noun(3),station)
GOBJ        =  inst(noun(2),instrumentation pod)
```

(d) Logical Form records derived from (c).

```
location(noun(253960-4-1),at(noun(253960-4-2))).
inst(noun(253960-4-1),HS camera).
location(noun(253960-5-2),at(noun(253960-5-4))).
inst(noun(253960-5-2),BTV).
location(noun(253960-6-2),at(noun(253960-6-3))).
inst(noun(253960-6-2),Skyray).
attribute(noun(253960-6-3),8).
inst(noun(253960-6-3),station).
attribute(noun(253960-5-4),7).
inst(noun(253960-5-4),station).
attribute(noun(253960-4-2),6).
inst(noun(253960-4-2),station).
```

By reducing all three stations to one, we can no longer indicate which piece of equipment is on which station. However, we can arrive at a solution by handling the IOBJ instances differently that the other objects. Recall that IOBJ's are those nouns that have a number or id following it. During the functional parse, we produce the records in Eg. 4.59(c). For IOBJ's then, we turn off anaphoric reference resolution and treat them as unique instances. The resulting logical form records are shown in Eg. 4.59(d).

While we have an easy solution for IOBJ's, we have no apparent easy solution for other objects where an identifier appears before the word. In Eg. 4.60, "pylon" in both sentences appears as a POBJ. On seeing the second occurrence of "pylon," we would attempt anaphoric reference resolution and maintain only one instance of "pylon" with both "outboard" and "inboard" modifying it. We are now forced to look at the adjectives in more depth and come up with a set of rules for indicating when we are dealing with two different objects. We conjecture that we would have to create classes of adjectives and that when two different adjectives from the same class precede two occurrences of the same noun, then we conclude that we have two distinct objects. If the adjectives are either both identical or of different classes, then we conclude that the two objects are one in the same. We have also left this as an unresolved issue.

Example 4.60. Resolving a series of identical nouns preceded by unique identifiers.

```
Sidewinder AIM 9R and Sidewinder AIM 9L mounted on F/A-18A BU# 162396
aircraft (nose 105 and tail 5 with flying eagle). AIM 9R on outboard pylon.
AIM 9L on inboard pylon. closeup front view.
```

In addition to anaphoric reference of nouns, we must also consider verbs. Examine the two occurrences of "load" in Eg. 4.61(a). The verb actions are one in the same. In the first sentence, we discover

Example 4.61. Resolving a series of identical nouns followed by unique identifiers.

(a) Anaphoric reference between underlined nouns in Caption 213855.

```
LGB Skipper bomb being loaded from MK 7 loader to A-7C BU# 156739 aircraft
(CL on tail). personnel loading.
```

(b) Functional Parse records for phrases containing underlined nouns in (a).

```
          Sentence 1

VPREP         =  from(pastpart(1),noun(5))
PREP          =  to(noun(5),noun(8))
OBJ           =  phys_obj(pastpart(1),noun(3))
MAIN-V        =  decl(pastpart(1),assemble)
ADJS-NOUN     =  bomb(noun(3),LGB)
ADJS-NOUN     =  bomb(noun(3),Skipper)
ADJS-NOUN     =  vehicle(noun(5),Mark-7)
ADJS-NOUN     =  aircraft(noun(8),A-7C)
POBJ          =  inst(noun(8),aircraft)
POBJ          =  inst(noun(5),loader)
GOBJ          =  inst(noun(3),bomb)
```

```
Sentence 2

SUBJ        =  human being(prespart(1),noun(1))
MAIN-V      =  decl(prespart(1),assemble)
GOBJ        =  inst(noun(1),people)
```

(c) Logical Form records derived from (b).

```
inst(noun(q-2-1),people).
destination(noun(q-1-5),to(noun(q-1-8))).
inst(noun(q-1-5),Mark-7).
inst(noun(q-1-3),Skipper).
inst(noun(q-1-8),A-7C).
theme(pastpart(q-1-1),obj(noun(q-1-3))).
source(pastpart(q-1-1),from(noun(q-1-5))).
agent(pastpart(q-1-1),subj(noun(q-2-1))).
activity(pastpart(q-1-1),assemble).
```

the *object*, *source*, and *destination* for the verb while in the second sentence, we discover the *agent*. The functional parse records are shown in Eg. 4.61(b). Notice that the semantic representation of "load" is "assemble" and that we have two different tokens: prespart(1) and pastpart(2). When mapping to the type hierarchy, we combine them into one token. Eg. 4.61(c) shows the resulting logical form records.

This situation where we combine the two tokens into one makes sense because we reason that the two events are one in the same. Eg. 4.62 shows a situation where this is not the case. If we use the same rationale as before, we would have a token for the first occurrence of "installed" and a second for the second occurrence. These tokens would be combined into one. The problem is that although we have "not" modifying the second "installed," it would apply equally to the first "installed" as well. We could handle this in one of three ways: treat "not" and "installed" as one verb and generate some verb besides "install;" replace the word with an "un-" prefixed version if one exists or some equivalent verb; or simply leave out the second sentence.

Example 4.62. Situation where we cannot combine verb instances (from Caption 238225).

NA-3B BU# 142360 Skywarrior aircraft with Standard Missile II-N and EX-62 TDD in nose for CFT. front view of plane with EX-62 <u>installed</u>. radome not <u>installed</u>.

The first approach is preferred as its solution would provide a uniform means to handle all such "not" cases. Attempts to modify the parse rules to accommodate this situation have been unsuccessful because of the multiple parse rules that applied. As a temporary solution, we have decided to leave out the sentence because of the unlikely situation of a query specifying some object that is not contained in the image. This temporary solution however is not necessary for all occurrences of "not." We currently handle "not" as an adverb. For those situations where we have a caption with either a single verb or a number of

unrelated verbs of which "not" is modifying one of them, we would simply handle "not" as an attribute of the verb it modifies. Hence, it is not necessary to remove all sentences that have a "not" preceding a verb.

4. Inferring Themes outside of Noun Phrases

As we saw above, *themes* for verbals are determined during noun phrase analysis. Another situation that involves inferring themes occurs when we have verbals that have no suitable object within the noun phrase. In the last sentence of the caption in Eg. 4.55(a) (and shown in Eg. 4.63(a)), we wish to find the theme for the verbal "air to air view." The approach we have taken is to examine the noun history list and find the most recent general object (GOBJ) noun that we encountered. The nouns "F3H-1," "BU#," "aircraft," and "tail" are all POBJs while "Point Mugu" is an ID object (IOBJ). The last GOBJ is "missiles" from the second sentence. Hence we infer "missiles" as the *theme* of "air to air view." In actuality, as a result of anaphoric reference on "missiles," the *theme* is "Sparrow 3." The logical form records produced are shown in Eg. 4.63(b).

Example 4.63. Finding the *theme* in the preceding sentence(s).

(a) Last sentence of Caption 10862.

```
air to air view from side during firing of one missile.
```

(b) Logical Form records for "view" theme.

```
inst(noun(10862-1-1),Sparrow 3).
theme(noun(10862-3-1),obj(noun(10862-1-1))).
inst(noun(10862-3-1),air-to-air view).
```

In actuality, we could argue that the theme should really be the "F3H-1" aircraft. However, we have found in the general case that this policy works a majority of the time. Unfortunately, the rule is not perfect.

Another example is provided by the verbal "flight" in the second sentence of Caption 900304 in Eg. 4.64(a). Looking back in the noun history list, the first GOBJ noun we encounter is "aircraft" which has been specialized to "B-2." Hence, the *theme* of "flight" is "B-2" as shown in Eg. 4.64(b).

Example 4.64. Finding the *theme* in the preceding sentence(s).

(a) Caption 900304.

```
air to air view of Northrop ATB (Advanced Technology Stealth Bomber) B-2
aircraft. first public flight in late 1989.
```

(b) Logical Form records for "flight" theme.

```
inst(noun(query-1-5),B-2).
theme(noun(query-2-1),of(noun(query-1-5))).
inst(noun(query-2-1),flight).
```

Looking at both of these last two examples, you may have noticed that both "F3H-1 aircraft" and "B-2 aircraft" were preceded by "of" and that in the former case, the noun was taken to be a POBJ while in the latter, it was a GOBJ. The reason is that "on underside of" is reduced to one preposition, "under," (with some loss of meaning) and that it, as well as some other prepositions, are treated differently than "of." Nouns following these prepositions are marked as POBJs while the nouns following an "of" are always marked as a GOBJs.

5. Inferring Verbs

We have come across just one situation where it seems that an adjective and preposition combined should infer a verb. Examine "clear of" in Eg. 4.65(a). We could rewrite this small phrase to read "missile cleared launcher" where "cleared" is a verb. The words "clear of" have been specially defined in the lexicon to indicate that an event has occurred. The functional parser then generates the records in Eg. 4.65(b).

"Free" is defined in the lexicon as the *fpcat* for "clear of." From these records, we infer the main verb to be "free" with "noun(3)" as the subject and "noun(4)" as the object. In mapping these records to the type hierarchy, we arrive at the following logical form records shown in Eg. 4.65(c).

Example 4.65. Inferring Verbs.

(a) Caption 166318.

```
Agile-Quickturn missile (FTV-2) leaving launcher. excellent view of missile
with plume. missile clear of launcher.
```

(b) Functional Parse records for phrases containing underlined words in (a).

```
PVERB   = free(noun(3),noun(4))
POBJ    = inst(noun(4),launcher)
POBJ    = inst(noun(3),plume)
```

(c) Logical Form records derived from (b).

```
agent(pverb(166318-3-a5),subj(noun(166318-1-2))).
object(pverb(166318-3-a5),obj(noun(166318-1-3))).
activity(pverb(166318-3-a5),free).
inst(noun(166318-1-3),launcher).
inst(noun(166318-1-2),FTV-2).
```

A related problem was one that became obvious when we were experimenting with the matching. Examine the caption in Eg. 4.66(a). The query stated "night vision goggles being worn." We would hypothesize that the "with" in the example could be construed as the act of "wearing" and that the query should match this caption. Eg. 4.66(b) shows the functional parse records for the underlined words.

To handled this situation, we added a rule when processing the "with" preposition that if we have a "human being" and "clothes" instances involved in the relationship, infer a "wear" event. The logical form records produced are shown in Eg. 4.66(c).

Example 4.66. Inferring Verbs.

(a) From Caption 258795.

```
Cdr Antonio with night vision goggles in cockpit.
```

(b) Functional Parse records for (a).

```
PREP          =  with(noun(2),noun(4))
PREP          =  in(noun(4),noun(5))
ADJS-NOUN     =  rank(noun(2),Commander)
ADJS-NOUN     =  abst_obj(noun(4),night vision)
POBJ          =  inst(noun(5),cockpit)
POBJ          =  inst(noun(4),goggles)
GOBJ          =  inst(noun(2),Antonio)
```

(c) Logical Form records derived from (b).

```
attribute(noun(258795-1-2),inst(noun(258795-1-a2))).
inst(noun(258795-1-2),Antonio).
inst(noun(258795-1-a2),Commander).
inst(noun(258795-1-a1),night vision).
attribute(noun(258795-1-4),inst(noun(258795-1-a1))).
location(noun(258795-1-4),in(noun(258795-1-5))).
inst(noun(258795-1-4),goggles).
agent(pverb(258795-1-a3),obj(noun(258795-1-2))).
theme(pverb(258795-1-a3),obj(noun(258795-1-4))).
activity(pverb(258795-1-a3),wear).
inst(noun(258795-1-5),cockpit).
```

6. Quantifiers

We have encountered several captions that use quantifiers such as "all," "some," "two," etc. Epstein (1988) proposed a method for handling them based on "principle-based grammars." We have implemented rules for handling simple situations, similar to the methods we saw in Section IV.F.3 for handling anaphoric reference. For cases where these rules are not effective, the mapping rules simply treat the quantifiers as adjective-quantifiers for the head noun that they modify. In Eg. 4.67(a), an AGM-88 Harm missile is loaded on all the previously mentioned aircraft, i.e., EA-6B, A-7E, A-6E, and F/A-18A. When the system encounters "aircraft," it will check to see if a universal quantifier is modifying it. If there is no modifier, aircraft is handled as a simple anaphoric reference to any previous aircraft that may exist. If there is a modifier, then we cache specific information indicating what objects are being universally quantified. Resolving the universal quantification, in this case removing the "aircraft" term and distributing slot values to the specific aircraft, is accomplished in the third pass of the logical form processing. The resulting logical form records are shown in Eg. 4.67(b).

Processing the universal quantifier in for this last caption was straightforward as we included all of the previous specific aircraft. Handling of an existential quantifier, however, may be the more difficult as we may not be able to know exactly which of several objects to include.

Example 4.67. Universal Quantification .

(a) From Caption 257274.

```
air to air view of four plane formation. clockwise from left to right: VX-
5''s EA-6B, NWC''s A-7E, NWC''s A-6E, and NWC''s F/A-18A. all aircraft
loaded with AGM-88 Harm. overhead right side view.
```

(b) LF records for "all aircraft loaded with AGM-88 Harm."

```
inst(noun(257274-2-8),F/A-18A).
inst(noun(257274-2-6),A-6E).
inst(noun(257274-2-4),A-7E).
inst(noun(257274-2-2),A-6B).
object(pastpart(257274-3-1),with(noun(257274-3-3))).
theme(pastpart(257274-3-1),obj(noun(257274-2-8))).
theme(pastpart(257274-3-1),obj(noun(257274-2-6))).
theme(pastpart(257274-3-1),obj(noun(257274-2-4))).
theme(pastpart(257274-3-1),obj(noun(257274-2-2))).
activity(pastpart(257274-3-1),assemble).
```

In Eg. 4.68, we have numeric quantifiers. In the second sentence, "three" modifies "F3D-1" after specialization while in the third sentence, "two" modifies "aircraft." As a result of anaphoric reference in the third sentence, our rules will replace "aircraft" by "F3D-1" with both "3" and "2" modifying it. We presently have no means of resolving the problem between "3" and "2."

Example 4.68. Numeric Quantification (in Caption 3204).

```
TP 1424. three F3D-1 aircraft on flightline. two aircraft carry Sidewinder
missiles.
```

A slightly different problem involves implicit quantifier scoping. Examine "view" in Eg. 4.69. We cannot be sure of how to handle "view" without looking at the image specifically. "Right side view" may refer to all three missiles, or it may only be associated with one of the objects. Our default rule connects it with the last object seen which is the "AIM 9C."

Example 4.69. Implicit Quantification (in Caption 29773).

```
Sidewinder 1A and Sidewinder 1C missiles. comparison of Sidewinder 1A AIM
9B, Sidewinder 1C IRAH AIM 9D, and Sidewinder 1C SAR AIM 9C. 3/4 right side
view on stand.
```

7. Relationship Chains

Relationship chains mostly occur when correlations are involved. Eg. 4.70(a) presents one of the cases. In the first sentence, the Sidewinder missile is located on the F/A-18C while in the second sentence we have that the missile is on the wing pylon. Through anaphoric reference resolution, notice that "AIM-9R" is located on both the "F/A-18C" and on the "pylon," In the type hierarchy, we have included

correlations to indicate that "wing" is *part_of* an "aircraft" and "aircraft" *has_part* a "wing." Hence a third pass of the logical form processing should check for these type of relationships and delete one of the two records, specifically, the one that states that the AIM-9R is located on the F/A-18C. If needed, we can determine this relationship later using the correlation. The logical form records involved in the chain are shown in Eg. 4.70(b).

Example 4.70. Relationship chains involving correlations.

(a) Caption 262868.

```
Sidewinder AIM 9R missile mounted on F/A-18C BU# 163284 aircraft (nose
110). closeup side view of missile on outboard wing pylon.
```

(b) Logical Form records for "flight" theme.

```
inst('noun(262868-1-3)','AIM-9R').
inst('noun(262868-1-6)','F/A-18C').
theme('pastpart(262868-1-1)',obj('noun(262868-1-3)')).
location('pastpart(262868-1-1)',on('noun(262868-1-6)')).
location('pastpart(262868-1-1)',on('noun(262868-2-4)')).
activity('pastpart(262868-1-1)',assemble).
attribute('noun(262868-2-a258)',outboard).
inst('noun(262868-2-a258)',wing).
attribute('noun(262868-2-4)',part_of('noun(262868-2-a258)')).
inst('noun(262868-2-4)',pylon).
```

We saw an example of a relationship chain previously in Eg. 4.66. If you examine the logical form records, you will see that the goggles are located in the cockpit. We also know that "Antonio" *wears* the "goggles." Hence, we could either add a rule in a third stage to infer that Antonio is in the cockpit as well or derive it using additional rules.

Other captions where relationship chains apply are shown in Eg. 4.71.

Example 4.71. Other captions containing relationship chains.

<u>ID</u>	<u>CAPTION</u>
43176	full side view of aircraft with dummy bomb <u>over</u> B-1-B target center.
43694	Rockeye II bombs <u>on</u> an A-6A aircraft ... bombs mounted <u>on</u> wings and centerline station.
252494	missile with large plume and <u>in front of</u> target.

G. POSTPROCESSING OF THE LOGICAL FORM

In the last section and during our discussion of the type hierarchy, we briefly discussed the desire for some additional processing of the logical form. During the matching experiments, several situations arose that made this desire a necessity. In the following two sections, we discuss the *transform* and *infer* postprocessing phases that were added to the system.

1. Transformations

We have seen how *themes* can be attached to verbals and we have discussed rules to create *themes* and *agents* for verbs by knowing whether they are passive or active. One of the areas of concern in creating the logical form deals with the handling of the verbals. Whenever we were able to parse them as nouns, processing of the adjectives and nouns preceding it went smoothly. Earlier in Eg. 4.23, we saw a situation where we could not force the parser to give us the noun form of the word, but only the verb.

Recall that "firing" was treated as a verb, specifically "launch." The verb sense appears feasible and would suggest that we treat all verbal forms as *activity's* instead of *inst's*. The problem is that "Shrike" is treated as an *agent* instead of *theme*. If we had a caption that began with "aircraft," then the case that is produced would be correct. In this case, however, we will have a problem.

Trying to match a query against the resulting logical form proved to be just as difficult. These problems and others drove the need to create a *transforms* slot and rules that could be applied to specific words if certain conditions were met. Transforms slots have the syntax shown in (37).

```
slot(Word, WordKey, transforms,                                        (37)
    [c(OldKase, OldReln, OldClass, OldKey,
       AssocKase, AssocReln, AssocClass, AssocKey,
       NewKase, NewReln, NewClass, NewKey)]).
```

The rule works as follows: if Word (sense identified by WordKey) has an OldKase (i.e., old *case*) record whose second argument is an instance of OldClass-OldKey enclosed in an OldReln relation predicate, and Word has an associate AssocKase record having an instance of AssocClass-AssocKey enclosed in an AssocReln, then transform the components of the OldKase record to components of a NewKase record. However, NewClass-NewKey refer to the new class for Word and WordKey.

Eg. 4.72 shows the simple case of transforming the noun "firing" (canonically represented as the noun "launch") into the verb "launch." A value of "_" in a transforms field will match anything.

Dick and Hirst (1991) brought out the point that verb-centered approaches provided a powerful representation of the information for retrieval purposes. Given that captions consist mostly of noun phrases, we believed that verbs would play a small role in retrieval. We saw in Eg. 4.65 how a verb

Example 4.72. Transformation involving the noun "firing."

(a) Examine "firing" in the third sentence of Caption 10862.

```
Sparrow III ... four missiles on underside of F3H-1 BU# 137010 aircraft ...
air to air view from side during firing of one missile.
```

(b) Logical Form records for "launch" before transforms rule applied.

```
inst(noun(10862-1-1),Sparrow 3).
theme(noun(10862-3-3),obj(noun(10862-1-1))).
inst(noun(10862-3-3),launch).
```

(c) Transforms rule for "launch."

```
slot(launch, noun-1, transforms,
    [c(inst, _, launch, noun-1,
       _, _, _, _,
       activity, _, launch, infin-1)]).
```

(d) Logical Form records for "launch" after transforms rule applied.

```
inst(noun(10862-1-1),Sparrow 3).
theme(noun(10862-3-3),obj(noun(10862-1-1))).
activity(noun(10862-3-3),launch).
```

(specifically "wears") could be inferred based on the ways nouns are connected via prepositions. We now believe that verbs could take on a more active role as we better understand the way relationships can be drawn between the nouns in nominal compounds and among prepositional phrases.

Transformations are not limited to verbals only. In Eg. 4.73, we have a transformation rule that takes an attribute case containing a *part_of* relation and translates it to location. Our goal in going from one case to another, from attribute to location in this example, is to obtain a more convenient representation to

Example 4.73. Transformation involving *part_of.*

(a) Examine "firing" in the third sentence of Caption 10862.

```
F/A-18 with Maverick missile.
```

(b) Logical Form records before *transforms* rule applied.

```
inst(noun(q-1-1),F/A-18)).
attribute(noun(q-1-3),part_of(noun(q-1-1)))).
inst(noun(q-1-3),Maverick).
```

(c) *Transforms* rule for "missile."

```
slot(missile, noun-1, transforms,
    [c(attribute, part_of, aircraft, noun-1,
       _, _, _, _,
       location, on, _, _)]).
```

(d) Logical Form records after *transforms* rule applied.

```
inst(noun(q-1-1),F/A-18).
location(noun(q-1-3),on(noun(q-1-1))).
inst(noun(q-1-3),Maverick).
```

inference additional facts. Also, having a location case as opposed to an attribute case may be beneficial during the matching when we encounter sentences that generate location records out right; an example is "Maverick missile on F/A-18" where we would normally generate a location record. We view the attribute record in particular as providing a stepping stone to achieving a better canonical representation. After going

through this exercise with the current system, a future modification we can make is to simply bypass the attribute record and just create the location record.

A slightly more complex transformation involves switching verbs and verb cases to arrive at a more basic canonical form for semantic representation. This type of operation is similar to some of the Conceptual Dependency mappings to the eleven basic ACTs. Eg. 4.74 shows the transformation involved in going from a "carry" event to an "assemble" event as we consider the meanings to be similar.

Example 4.74. Transformation involving *verbs and verb cases..*

(a) Transformation involving "carry" from the third sentence of Caption 252492.

```
F/A-18A carrying two Sidewinder AIM 9M's.
```

(b) Logical Form records before *transforms* rule applied.

```
quantity(noun(q-1-3),plural(2)).
inst(noun(q-1-3),AIM-9M).
inst(noun(q-1-1),F/A-18A).
agent(prespart(q-1-1),obj(noun(q-1-1))).
theme(prespart(q-1-1),obj(noun(q-1-3))).
activity(prespart(q-1-1),carry).
```

(c) *Transforms* rule for "missile."

```
slot(carry, infin-1, transforms,
    [c(agent, _, aircraft, noun-1,
       theme, _, missile, noun-1,
       location, on, assemble, infin-1)]).
```

(d) Logical Form records after *transforms* rule applied.

```
quantity(noun(q-1-3),plural(2)).
inst(noun(q-1-3),AIM-9M).
inst(noun(q-1-1),F/A-18A).
theme(prespart(q-1-1),obj(noun(q-1-3))).
location(prespart(q-1-1),on(noun(q-1-1))).
activity(prespart(q-1-1),assemble).
```

2. Inferring Additional Facts

The inference procedures for this phase are similar in scope to the *transforms* rules we saw in the last section. The rules attached to specific concepts look to see what is known with respect to a concept, and if something does not exist, then it is added. These rules are often domain-dependent and require additional manual effort. Our reason for including them is that we encountered situations where this type of information was needed during matching based on the test queries. This need and ensuing implementation begin to suggest the use of scripts in matching, something we wished to avoid at the start of this work in attempting to derive a neutral representation. The general format of the *infer* rule is:

```
slot(Word, WordKey, infer,                                        (38)
     [c(Kase, Reln, Class, Key,
        InferKase, InferReln, InferClass, InferKey)]).
```

This rule works as follows: if Word has a Kase record whose second argument is an instance of Class-Key enclosed in an Reln relation predicate, then we can infer a new InferKase record whose second argument is an instance of InferClass-InferKey enclosed in an InferReln. To illustrate, in Eg. 4.72, the concept "air to air view" was used. Its *theme*, which is not stated explicitly, is inferred to be the "Sparrow 3" missile. However, we also know that the location of the Sparrow 3 must be in the air because we have an "air to air view." Eg. 4.75 shows the logical form records, before and after the inference, and the *infer* rule applied.

Example 4.75. Inference involving "air to air view" from Eg. 4.72.

(a) Logical Form records before *infer* rule applied.

```
theme(noun(10862-3-1),obj(noun(10862-1-1))).
inst(noun(10862-3-1),air-to-air view).
inst(noun(10862-1-1),Sparrow 3).
```

(b) *Infer* rule for "carry."

```
slot('air-to-air view', noun-1, infer,
    [c(theme, _, _, _,
       location, in, air, noun-1)]).
```

(c) Logical Form records after *infer* rule is applied.

```
theme(noun(10862-3-1),obj(noun(10862-1-1))).
inst(noun(10862-3-1),air-to-air view).
location(noun(10862-1-1),in(noun(10862-3-a3))).
inst(noun(10862-1-1),Sparrow 3).
inst(noun(10862-3-a3),air).
```

A similar type approached was described by Finin et al. (1991) in which new facts are added to the database if certain phrases are found to exist in the text being parsed. The example described is that a terrorist organization should be inferred and a fact cached to the database if the text is found to contain the string "Army of National Liberation."

A somewhat riskier inference is shown in Eg. 4.76 involving the verb "uploaded." We see that several pieces of equipment are "uploaded," but the destination is not explicitly stated. However, we can

Example 4.76. Inference involving "uploaded."

(a) Caption 231373.

```
air to air view of Harm missile launched from F/A-18A BU# 161720 aircraft
(nose 102) over NWC ranges. Harm, pod, and MK 91 Silver Bullet uploaded.
excellent view of aircraft just before firing of missile.
```

(b) Logical Form records with respect to "uploaded" before *infer* rule applied. Note that "uploaded" is being mapped to "assemble."

```
theme(pastpart(231373-2-1),obj(noun(231373-1-3))).
theme(pastpart(231373-2-1),obj(noun(231373-2-2))).
theme(pastpart(231373-2-1),obj(noun(231373-2-4))).
activity(pastpart(231373-2-1),assemble).
inst(noun(231373-2-4),MK-91).
inst(noun(231373-2-2),pod).
inst(noun(231373-1-3),Harm).
```

(c) *Infer* rule for "missile," "bomb," and "pod."

```
slot(missile, noun-1, infer,
    [c(activity, _, assemble, infin-1,
       location, on, aircraft, noun-1)]).
slot(bomb, noun-1, infer,
    [c(activity, _, assemble, infin-1,
       location, on, aircraft, noun-1)]).
slot(bomb, noun-1, infer,
    [c(activity, _, assemble, infin-1,
       location, on, aircraft, noun-1)]).
```

(d) Logical Form records after *infer* rule is applied.

```
inst(noun(231373-1-6),F/A-18A).
location(pastpart(231373-2-1),on(noun(231373-1-6))).
theme(pastpart(231373-2-1),obj(noun(231373-1-3))).
theme(pastpart(231373-2-1),obj(noun(231373-2-2))).
theme(pastpart(231373-2-1),obj(noun(231373-2-4))).
activity(pastpart(231373-2-1),assemble).
inst(noun(231373-2-4),MK-91).
inst(noun(231373-2-2),pod).
inst(noun(231373-1-3),Harm).
```

implicitly make the assumption that it is on the aircraft. Hence, we can add a rule that makes the same inference.

H. SUMMARY

This chapter has discussed the major processes and problems associated with creating a neutral logical form. One of our subgoals in using the DBG parser was to be able to apply this system for processing not only typed-in captions and queries, but spoken queries as well. The transformation and inference rules we have just seen are driving us towards formalizing the use of frames and scripts for knowledge representation in order to increase the effectiveness of the matching. Both the representation and the rules we have used require manual creation by a knowledge engineer or retrieval specialist. We originally speculated that the type hierarchy would be one dimensional in the sense that classes (or concepts) would simply be known by their position in the hierarchy. We can view the type hierarchy as an extension of the lexicon. Through the *correlations* and *infer* slot, we have a rudimentary ability to indicate how concepts are connected and interact with one another which can be an important part of the lexicon. This concept coherence notion has been

suggested by Alterman (1985) who defines seven binary relations of which class/subclass is one. These relations, though, pertain only to events and states. However, slots could be added to specify newer relations for verbs and nouns as the quality of knowledge required increases.

The extent to which these types of enhancements must be implemented, however, is still uncertain. As you will discover in the next chapter, the types of queries we have encountered have not required the more rigorous level of detail. However, we surmise that as the system is used, the user will expect more accurate responses and such structures may need to be introduced.

V. CAPTION MATCHING

This chapter describes the process by which a query logical form is matched against the caption logical forms.

A. INTRODUCTION

To enhance the usability of this system, we wanted the users to be able to enter queries using natural language phrases or sentences. Anick (1991) pointed out the user preference for NL querying versus queries consisting of Boolean expressions when looking at enhancements for the STARS retrieval system. Hence, we decided to use the same parser for both parsing captions and queries. This approach contrasts with the one taken by Mauldin (1991) where a text scanner is used to process the documents and CD type structures are used for queries. Once the logical form records for the query are produced, the caption matching process proceeds in two phases similar in nature to the approach taken by Rau (1987). The first phase is an enhanced keyword search using a type hierarchy to find specializations of keywords. We refer to this phase as a *coarse-grain* match. The second phase entails graph matching where both the query and caption logical forms are viewed as graphs and we are attempting to determine if the caption graph may contain a subgraph of the query graph. This latter phase is termed the *fine-grain* match.

Both the coarse- and fine-grain matches require some data repository of caption information to perform matching against. This data repository is found in the *semantic database*. The database consists of keyword files reflecting class concepts found in the type hierarchy. These keyword files are used exclusively by the coarse-grain match. The actual logical form records provide the second portion of the semantic database and they are used exclusively by the fine-grain match.

The VISUAL keyphrase retrieval system used at the NAWCWPNS China Lake Photo Lab (keyphrase matching) will be briefly described to provide some background upon which our match process is developed. We then describe both the coarse- and fine-grain match phases respectively.

B. KEYPHRASE MATCHING

Recall from Chapter III that the image registration data for MARIE is stored in the *visual* relation. Retrieval of this can be done by formulating an SQL query statement. If an index exists on an attribute, the index will be used. The caption field is not indexed. Searching on the caption field requires creating a regular expression and defaults to sequential search. To improve performance, the retrieval specialist manually creates *keyphrase* records for either a single or set of photographs. Each keyphrase consists of a head keyword and a string of descriptors. Some of the keyphrases currently used are shown in Table 5.1.

TABLE 5.1. KEYPHRASE RECORD FORMATS.

Keyword	Descriptors
AIRCRAFT	Designator / Squadron or nothing is NAWCWPNS, CL (it is understood that a/c is complete, if no list partial or cockpit/project name/what (if applicable (firing, drop etc.)))
CHEMISTRY	Project Name / What / Where
FUZE	Project Name / Designator / What TDD, S&A, EA, EX, ignitor, detonator, destructor.
GEOLOGICAL	What / Where (Aerial if applicable)
MISSILE	Project Name / What / Where (No SNORT info)
OPTICAL	Item (Acquisition)
PROJECTILE	Project Name / Size (use " for inches) / What / Where
TARGET	What Kind of Place / Project Name

Notice that each keyphrase has a certain format with rules as to what can be entered. The keyphrase records are recorded in the *keyphrases* relation where the relationship between *visual* and *keyphrases* is one to many (1:M). Hence, one visual registration record can have multiple keyphrase records. Table 5.2 shows the keyphrase records for captions 262865 through 262873 (see Appendix B for the caption descriptions).

TABLE 5.2. KEYPHRASE RECORDS FOR CAPTIONS 262865-73.

Id	Keyphrase
262865-73	AIRCRAFT F/A-18C PARTIAL SIDEWINDER AIM-9R AF
262865-73	AIRCRAFT F/A-18C SIDEWINDER AIM-9R AF
262865-73	MISSILE SIDEWINDER AIM-9R F/A-18C AF
262865-73	MISSILE SIDEWINDER AIM-9R STAND AF
262865-73	PROGRAM SIDEWINDER AIM-9R

Although systems now exist for automatically creating keyphrases [Driscoll 91] and traversing keyphrase hierarchies [Ragusa 90], their effectiveness is still limited. Nevertheless, the Photo Lab retrieval specialist currently must analyze the captions written by the photographer and manually create the keyphrases. Our approach entails MARIE constructing the index and match constructs directly from the captions. Thus instead of the retrieval analyst creating new keyphrase records for each image (or image set), he needs only update the lexicon and type hierarchy for new words and concepts that previously did not exist. The caption, though, still needs to be written.

Although the captions form the primary source of input for our system, the keyphrase records may contain additional information not present in the captions. We have had to make the assumption that if there was additional information added, then that information should have been placed in the captions as well.

C. COARSE-GRAIN MATCHING

Coarse-grain matching is an enhanced keyword matching approach based on class/subclass ordering as specified in the type hierarchy. Instead of looking for captions that contain just the class (or keyword), we look for captions that contain the subclasses as well. This section describes how the index keys are determined, stored, and matched within the semantic and relational databases.

1. Semantic Database - Keyword Index Files

The semantic database as a whole is organized by the various multimedia categories; image, graphic, sound, text, and video data form the five subdivisions. Within each subdivision, the keyword index files are divided into nouns and verbs as there could arise situations where a word can be interpreted as either a noun or verb.

Let N be the set of noun instances present in a caption logical form and let C_N be the set of noun classes corresponding to the elements of N (recall that C_T is the set of all classes in the type hierarchy T). Hence $C_N \subseteq C_T$. V and C_V are defined respectively for the verbs. The set of instances, I, for a caption is defined as $N \cup V$. Eg. 5.1 illustrates these sets. The set $C_I = C_N \cup C_V$ identifies the keyword index files that need to be updated in the semantic database for a new caption.

Example 5.1. Caption 183531 (consult Appendices B and F).

```
N  =  {noun(183531-1-6), noun(183531-3-2), noun(183531-1-a45), noun(183531-1-7)}

CN =  {'PCH-1', 'side view', 'Harpoon', 'Puget Sound'}

V  =  {prespart(183531-2-1)}

CV =  {launch}
```

In order to update the keyword index files, update records are generated for each instance depending on its use (i.e., its *case*) in a caption. The default case for a keyword is *inst* for nouns and *activity* for verbs. It is conceivable that a keyword can appear in more than one case in a caption. To illustrate, in the logical form of Caption 183531, 'PCH-1' appears as the *theme* of "side view" and the *agent* for "launch." A default case is mapped to more specialized case if one exists and additional specialized cases are added to a list (see Eg. 5.2).

Example 5.2. Keyword Update Records for Caption 183531.

```
key_index('Harpoon',noun,[183531-theme]).
key_index('Puget Sound',noun,[183531-location]).
key_index('side view',noun,[183531-inst]).
key_index('PCH-1',noun,[183531-theme,183531-agent]).
key_index(launch,verb,[183531-activity]).
```

Each keyword index file has records containing the *caption identifier* and *case*. The rationale for including the case information in the index is to provide a mechanism to be more selective when performing the coarse-grain match. For example, when a query is entered, the user could specify the role for a word (i.e., its case). We can then use this information as a filter in selecting the applicable keyword records. Thus, if we were interested in those captions that have an F-4 aircraft as the *destination* of a missile launch, we would want to filter out those captions where the F-4 aircraft was actually firing a missile (*agent*) or merely being a place for some other event (*location*).

2. Determining the Match Criteria

For the coarse-grain match, we need to define a level of performance to determine which captions show promise for further analysis. This amounts to determining which keywords are to be searched for and what threshold value, k, should be set to filter out those captions which contain a less than adequate number of keywords. Let C_I be the set of classes (keywords) corresponding to the instances of the query and $\Phi(C_I)$ be a function of C_I that selects a keyword subset C_q for matching. In addition, let $U = D_1, D_2, ..., D_n$ be the collection of class subsets such that D_i consists of all the classes for the instances of caption$_i$ for the n captions in the database (eg., the classes for Caption 123456 are found in D_1, ...). The coarse-grain match problem is to determine the set of caption identifiers, M, corresponding to those D_i in U such that there exists a $Q = C_q \cap C_i$ where $\Phi(D_i)=C_i$ and $|Q| \geq k$. An initial approach might be to set $C_q = C_I$ and $k = |C_q|$. Eg. 5.3 illustrates the concept.

Example 5.3. *Query* = Sidewinder mounted on a stand.

C_N = {'Sidewinder', stand}
C_V = {assemble}

C_q = C_I = {'Sidewinder', assemble, stand}
k = 3

For each $x \in C_I$, let $\Psi(x)$ be a function that collects all of the subclasses of x into a set. Any candidate caption must have at least k keywords (y_1, y_2, ... y_k) where $y_i \in C_I$, or $y_i \in \Psi(x)$. Superclasses for each $x \in C_I$ need not be examined because the user would have specified a more general term if one was desired. For example, if we replace "Sidewinder" with "missile" in a query, then we should be able to find any missile mounted on a stand. By a user explicitly stating "Sidewinder", the match is constrained to looking at only the"Sidewinder" keyword index files and the various subclass (version) index files. The remainder of this section discusses issues in defining the function Φ.

a. Treatment of Verbs

Suppose that we were matching the query in Example 5.3 against Caption 262865 ("3/4 front view of Sidewinder AIM-9R missile on stand") where $C_{I(262865)}$ = {'front view', 'AIM-9R', stand}. A

match would have Q = {'Sidewinder', stand} where 'Sidewinder' $\Leftrightarrow$ 'AIM-9R' and stand $\Leftrightarrow$ stand. Caption ID 262865 $\notin M$ since $|Q| = 2 < k = 3$. The coarse-grain match would fail to consider this caption even though semantically they can be considered to be the same. Hence, we hypothesize that k should be less than C_I and that it should be based on the number of noun classes (C_N) to ensure that the major noun concepts in the query are found in a caption. A working hypothesis that we made was to not include verbs in the coarse-grain match.

b. *Treatment of Nouns*

The end-result of our coarse-grain match is a list of those caption identifiers, M, that have at least k query noun keywords contained in the caption. However, this is not always the case. In Eg. 5.4, we literally ask to find those captions that contain "type" as well as "Sidewinder." In actuality, we want

Example 5.4. *Query* = types of Sidewinder missiles.

$C_q = C_I =$ {'Sidewinder', type}
$k = 2$

multiple distinct Sidewinder or AIM-9*x* missiles. We have a semantic problem in applying our keyword lookup. Queries involving "kind of," "version of," "instance of," etc. pose the same problem. Words like "type" appear in only a few captions. This phenomenon would lead us to believe that we could filter them out when compared to words like Sidewinder or missile. Information theory suggests that we could assign more weight to certain words and base the search on total weight [Abramson 63]. However, certain nouns (e.g., "ACIMD" and "bomb skid") appear in a small number of captions as well as "type," and are better keyword discriminators than "type."

We see a similar problem when we deal with words that are indicative of the form of multimedia. For example, if we are interested in retrieving images, then expressions like "view of," "picture of," "photograph of," etc. add no value to the match criteria. They only serve to increase the number of keywords that must be found in a caption. We conclude that we must introduce an exception list, E, of general terms applicable across all multimedia types and remove these noun keywords from the list of nouns to be used in the match.

c. *Defining the Function Φ*

The function Φ can be considered our coarse-grain match filter. It will select those classes (keywords) that will be used in the match. For any query where C_I is the set of classes (keywords) in the query and E is the exception list, Φ is defined as:

$$\Phi(C_I) = C_I - C_V - E = C_N - E = C_q \quad (1)$$

We set $k = |C_q|$ as we hypothesize that this provides a lower bound on the number of keywords a caption should have in common with the query.

3. Matching using the Semantic Data

We begin our discussion of the matching process by defining some basic sets that we will need. For $x_i \in C_q$ where $1 \leq i \leq k$, let $T_i = \{x_i\} \cup V_i$ where $\Psi(x_i) = V_i$. Also, let $K = \{x : x$ corresponds to the identifier for caption$_i\}$. We define a new function $\Omega(y)$ where $y \in T_i$ and Ω returns the set of caption identifiers contained in the keyword file y. Recall that it is possible for a caption identifier to appear more than once in a keyword file if the keyword was used in more than one *case* in the caption. We are only checking for existence at this point. A future enhancement would allow returning a caption identifier only if it is connected with a specific *case* (e.g., *actor*, *location*, etc).

The coarse-grain matching process can be described as returning the set M where $M = \{<x, n> \in K \times P$: for $y \in T_i$ where $1 \leq i \leq k$, if $x \in \Omega(y)$, then $n = n + 1\}$ (P is the set of positive integers and n is initially set to 0). We can consider M as the result of a modified *union* set operation involving the caption identifiers associated with each query keyword, where x corresponds to a caption identifier and n represents the number of times x was seen during insertion into M; we refer to this operation as the *union-count* of sets. Note that M includes only those caption identifiers containing any of the keywords appearing in a T_i. Eg. 5.5 demonstrates the concept.

Example 5.5. Coarse-grain matching using query from Eg. 5.3.

```
Cq = {'Sidewinder', stand}
k = 2

T1 = {'Sidewinder', 'AIM-9', 'AIM-9B', 'AIM-9C', 'AIM-9D', 'AIM-9L', 'AIM-
     9M', 'AIM-9R', and 'modified AIM-9C'}
T2 = {stand, pallet, pedestal, rack, 'bomb skid'}

Ω('Sidewinder')= {3204, 5824, 258795}
Ω('AIM-9') = {251701, 251703, 251704, 251706, 251707}
Ω('AIM-9B') = {29773}
Ω('AIM-9C') = {29773}
Ω('AIM-9D') = {29773}
Ω('AIM-9L') = {161082, 182711, 182712, 182713, 242099, 256393, 256394,
               256395}
Ω('AIM-9M') = {216382, 252492, 252494, 252496, 255577, 255578, 255580}
Ω('AIM-9R') = {247181, 247186, 256393, 256394, 256395, 257055, 262865,
               262866, 262867, 262868, 262869, 262870, 262871, 262872,
               262873, 264968}
Ω('modified AIM-9C') = {221353, 221354}
Ω(stand) = {29773, 228544, 228545, 228546, 234064, 262865, 262866, 262867}
Ω(pallet) = {181761}
Ω(pedestal) = {}
Ω(rack) = {34070, 239097}
Ω('bomb skid') = {10851}
```

```
M =  {<29773,2>, <262865,2>, <262866,2>, <262867,2>, <3204,1>, <5824,1>,
      <10851,1>, <34070,1>, <161082,1>, <181761,1>, <182711,1>, <182712,1>,
      <182713,1>, <216382,1>, <221353,1>, <221354,1>, <228544,1>,
      <228545,1>, <228546,1>, <234064,1>, <239097,1>, <242099,1>,
      <247181,1>, <247186,1>, <251701,1>, <251703,1>, <251704,1>,
      <251706,1>, <251707,1>, <252492,1>, <252494,1>, <252496,1>,
      <255577,1>, <255578,1>, <255580,1>, <256393,1>, <256394,1>,
      <256395,1>, <257055,1>, <258795,1>, <262868,1>, <262869,1>,
      <262870,1>, <262871,1>, <262872,1>, <262873,1>, <264968,1>}
```

As the set and example show, the caption score, n, is a positive integer where $1 \leq n \leq k$. However, there is an inherent flaw with the set implementation stemming from the condition "if $x \in \Omega(y)$." Returning to Eg. 5.5, let $x = 29773$ and $\Omega(y)$ return the caption identifiers for each element of the set T_1. Notice that $29773 \in \Omega$('AIM-9B'), Ω('AIM-9C'), and Ω('AIM-9D'). Should n be incremented thrice or once? Should Caption 29773 carry more weight than a caption that has only one occurrence of a Sidewinder missile? We argue that for this caption to be preferred over one with only one Sidewinder term, the query should explicitly state that a plural number of missiles is desired. However, the implication of a plural noun (e.g., "Sidewinders") could be interpreted as a search for either multiple missiles with different versions (as in Caption 29773) or multiple missiles of the same type ('AIM-9Cs'). The former situation can be handled by the current keyword file structure. The latter situation requires that we include an additional field in the keyword record to indicate whether the keyword used in a caption is either singular or plural. We will not pursue the plural problem at this time. Hence, we will only check for the first occurrence of a caption identifier among the caption identifiers associated with the elements of any particular T_i. The set M can now be redefined as $M = \{<x, n> \in K \times P$: if there exists a $y \in T_i$ where $1 \leq i \leq k$ and $x \in \Omega(y)$, then $n = n + 1\}$.

We now define the *union-count* operation in terms of a two step algorithm. We must first create an intermediate set, Z, which contains the simple union of all the caption identifiers returned by $\Omega(y)$ where $y \in T_i$; this will prevent us from counting a caption identifier twice for $keyword_i$ where $keyword_i \in C_q$. The next step involves creating the ordered pair $<x, 1>$ for each $x \in Z$ and inserting it into the set M (initially set to $\emptyset$) such that if an ordered pair $<x, n>$already exists in M, then we replace $<x, n>$ with $<x, n+1>$; otherwise we simply insert $<x, 1>$ into the set. This two step algorithm must be performed for each $keyword_i$ in C_q.

4. Matching using the Relational Data

The matching operation performed in the semantic database is a straightforward keyword lookup using a collection of keyword index files. We can also perform semantic analysis on the query at this time to determine if some of the query information might reference the registration data. For example, if a query mentions a location, we have a mapping function that will examine the *location* field in the *visual* relation using an SQL *select* statement as shown in Eg. 5.6.

Example 5.6. Referencing the Relational Database (registration data).

```
Query = Maverick missile at Armitage Field.

    select id
    from visual
    where location = "Armitage Field"
```

The list of *identifiers* returned is a set since *id* is the primary key. Similar analysis is done for dates which map to the *date_orig* field. Multiple expressions in the *where* clause are discussed in the next chapter.

Let us define a new function *sql(y)* where $y \in T_i$ and *sql* returns the set of caption identifiers associated with the attribute value y in the *visual* relation. The set M can now finally be defined as $M = \{<x, n> \in K \times P$: if there exists a $y \in T_i$ where $1 \leq i \leq k$ and ($x \in \Omega(y)$ or $x \in sql(y)$), then $n = n + 1\}$. We modify our earlier definition of the *union-count* algorithm to include the function *sql* by performing a set union of the caption identifiers returned by *sql* with the intermediate set Z for $keyword_i$ in the first step; the second step is unchanged. Including the results of *sql* into Z can be done any time prior to initiation of the second step. The problem of defining a multi-argument *sql* function and incorporating its results into the *union-count* algorithm are discussed in the next chapter.

D. FINE-GRAIN MATCHING

The fine-grain match entails matching the query logical form against the caption logical form. The type hierarchy provides the framework to accomplish the matching operation. Both the query and caption logical forms can be viewed as directed graphs, $G_q = (V_q, E_q)$ and $G_c = (V_c, E_c)$ respectively, where the vertices represent the noun and verb instances and the edges represent the *case* relations.* Let n_q and n_c be the total number of vertices and edges in G_q and G_c respectively, i.e, $n_q, = |V_q| + |E_q|$ and $n_c = |V_c| + |E_c|$. We begin by making the hypothesis that the maximum possible score for a match is n_q. The problem is to determine if G_c contains a subgraph isomorphic to G_q.

Let m be the match score, i.e., the total number of vertices and edges that G_q and G_c have in common. From the coarse-grain match and our assumption above, we know that $k \leq m \leq n_q$. Let's now examine some of the possible values for m. First, if $m = k$, this indicates nothing more than a keyword match. Second, if $k < m = n_q$, then we have an exact match of the query against (possibly part of) the caption, i.e., we have found a subgraph of G_c that is isomorphic to G_q. Lastly, if $k < m < n_q$, we have found a subgraph of G_c that is isomorphic to a subgraph of G_q and we have a partial match of the query

* Since each vertex is also a node in the type hierarchy, we have the appearance of a three-dimensional structure.

against the caption. We do not have an exact solution to the problem at this point, only an indication of how far we have gotten.

This last condition above requires further analysis. There are two possible situations that can arise: $m = n_c$ and $m < n_c$ (note that these can also occur when m = n_q, but then we have an exact match). The situation where $n_c < m$ cannot possibly occur. For the former case ($n_c = m < n_q$), the query contains additional information over the caption (see Eg. 5.7). Note that there are actually less records in the caption to match against the query. The question arises as to whether we can consider this to be an exact match. We believe that it is. For the latter case ($m < n_c$ and $m < n_q$), we can consider this to be a true partial match.

Example 5.7. Match condition where $n_c = m < n_q$.

```
Query = missile mounted on F/A-18 aircraft

    inst(noun(query-1-1), missile).
    inst(noun(query-1-2), 'F/A-18').
    activity(pastpart(query-1-1), assemble).
    theme(pastpart(query-1-1), obj(noun(query-1-1))).
    location(pastpart(query-1-1), on(noun(query-1-2))).

Caption = Sidewinder AIM-9M on F/A-18A

    inst(noun(123456-1-1), 'AIM-9M').
    inst(noun(123456-1-2), 'F/A-18A').
    location(pastpart(123456-1-1), on(noun(123456-1-2))).
```

For both of these cases, we need a manner in which to present the exact and partial match answer to the user. For example, this query may have matched a different caption with a score of five, whereas the match score for this example would be three. A possible approach is to present user results not on the number of records that matched, but on the difference between the the number that matched and the number of records in the query. For example, if m - $n_q \leq 0$, then we have an exact match, otherwise we have a partial match that missed by the difference m - n_q. Thus, we could display search results to a user in the range [0, m - n_q] where 0 indicates an exact match and m - n_q indicates the number of records that failed to match. These or other measures can be easily implemented after the Photo Lab has more experience with our system and a suitable user interface is selected.

Figure 5.1 shows the relevant type hierarchy and graph structure for Eg. 5.7. We will study this match shortly, but first we describe how the edges are handled. We consider some of the edges as consisting of two parts, the *case relation* (e.g., location) and a *case relation modifier* (e.g., on), and others as consisting of just the case relation. In matching a query edge against a caption edge, we first check for the existence of like case relations. If they exist and are found to be related to query and caption instances being matched, then we have a basis upon which to assign scores for deriving the match score *m*. We consider the edge weight as consisting of two fractional parts whose sum is 1.0; the case relation modifier provides one-half and the

destination node instance provides the other half. This allows us to assign fractional scores to edges and hence have a fractional match score. For a case relation modifier, we can check for exact matching (e.g., on ⇔ on) or similarity matching (e.g., on ⇔ at). We hypothesize that the similarity matching requires further analysis on how the class endpoints of the edge may interact. We have implemented similarity matching moderately with respect to the prepositions "on," "at," and "under."

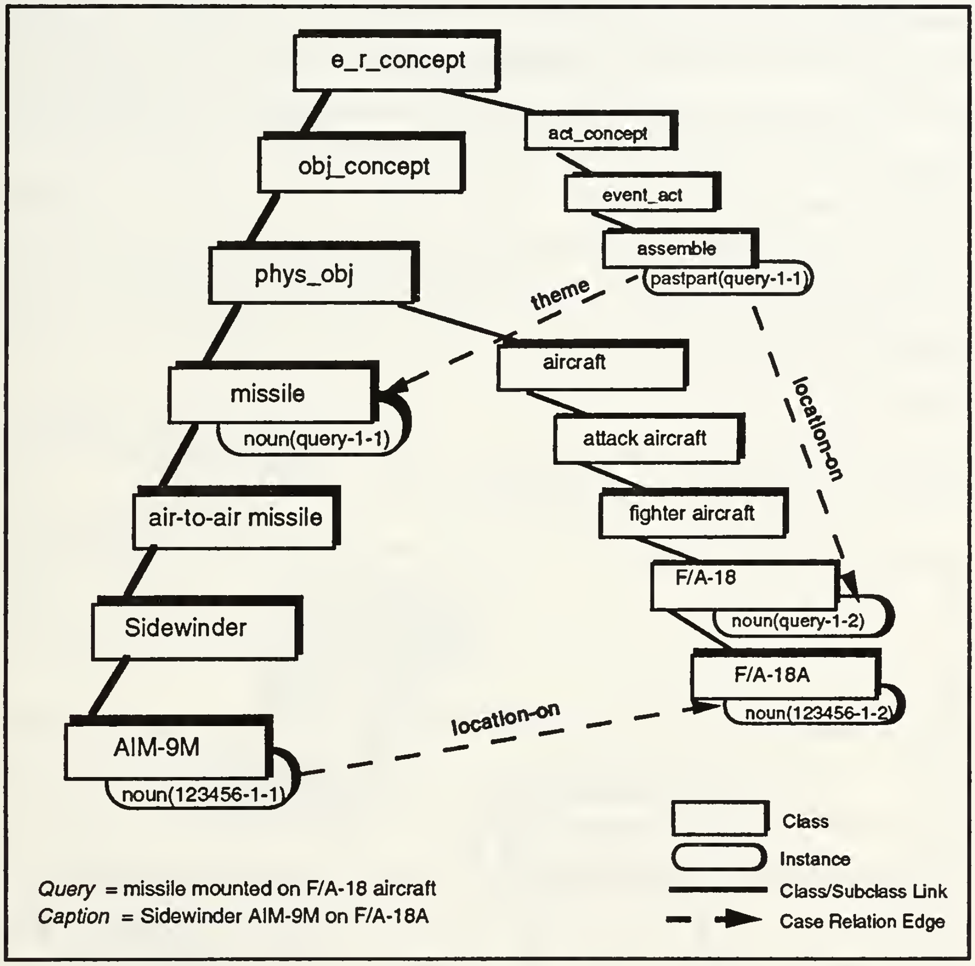

Figure 5.1. Type Hierarchy & Graphs for Eg. 5.7.

The fine-grain matching process is divided into four steps: (1) instance matching, (2) direct case relation matching, (3) indirect case relation matching (phase 1), and (4) indirect case relation matching (phase 2). The fine-grain match is the performance bottleneck for MARIE and we will make some estimates on the time spent for each based on experimental tests conducted. Each of these steps will now be discussed.

1. Instance Matching

Instance matching involves finding the corresponding vertices in the query and caption that can be matched. The vertices are defined by the *inst* and *activity* case relations. The vertices are matched based on class/subclass relationships, i.e., a query instance will match a caption instance if the caption instance class is either the same as or a subclass of the query instance class. In Figure 5.1, the query instance for the class "missile" matches the caption instance for the class "AIM-9M;" other classes are handled similarly. Hence, the query instances form a cover over the caption instances and the matching proceeds downward, from the query instance nodes to the caption nodes. We do not attempt to sort query instances based on some selective criteria as we believe it adds only extra processing time to the match because all query instances need to be examined sooner or later.

Occasionally, we may have multiple caption classes that could match one query class or vice versa. For example, consider the query "a view of a missile" (before we started filtering out "view" for images) under match with Caption 5824 (see below). "View" could match "air to air view" or "side view" as Figure 5.2 shows.

Caption 5824. air to air view of F3H-1 BU# 133550 aircraft with China Lake on tail. full side view with Sidewinder missile aboard aircraft.

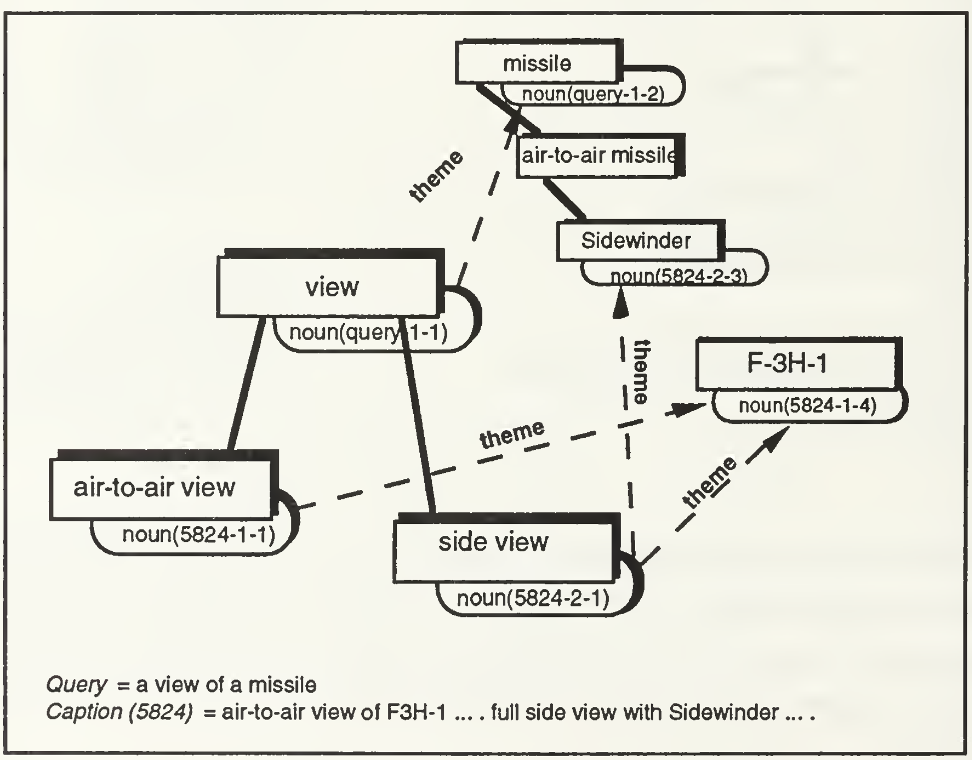

Figure 5.2. Matching "view" in Query with Caption 5824.

To handle this problem, we inserted a history list to indicate which query-caption instance combinations were previously attempted. This history list can be used to control either a backtracking or a recursive-call strategy (as an implementation note, we simply opted for the recursive-call method or pass instead of a repeat-fail loop for backtracking). We then attempted to find a new combination. If at least one new combination was found, we proceeded with the next three steps. If we had no new counterpart for a query-caption instance combination, then we used a match combination from a previous pass to continue (e.g., "missile" and "Sidewinder"). This selection of a previous instance has the potential for a less than optimal score (see Eg. 5.8). If we now go back and examine the caption more closely, we could obtain a match score of 4 if we used the combination "view ⇔ side view" and "missile ⇔ NWC Sidewinder"

Example 5.8. Matching using Multiple Passes.

```
Caption =    air to air view of F3H-1 ... full side view with NWC
             Sidewinder and USAF Maverick.
Query =      view of NWC missile.

Pass 1:
   (1) view ⇔ air-to-air view
   (2) missile ⇔ NWC Sidewinder
   ⇒ m = 2 (inst's alone)
Pass 2:
   (1) view ⇔ side view
   (2) missile ⇔ USAF Maverick
   ⇒ m = 3 (inst's + theme)
Pass 3:
   No untried instances remaining, stop.
```

(inst's + theme + attribute). To solve this problem, we must generate all of the possible instance match combinations before this first step and then examine each combination in a separate pass.

For each pass, we record the match score for the query-caption instance combinations. When we have exhausted all of the combinations and steps for each combination, we terminate the matching for the query and caption, and select the maximum score from all of the passes. This maximum value is then the score returned to the user for the caption. We estimate the time spent on instance matching is roughly 15% of the total fine-grain match time.

2. Direct Case Relation Matching

Direct case relation matching involves checking each query-caption instance combination to determine if they have matching slot values. For any LF record besides *inst* and *activity*, these slot values correspond to the case relation name and second argument of the record. Both the case relation name and second argument are stored in the *case_val* slot of an instance. Refer to Section IV.F.2 for a list of the various types of *cases*. For some *cases*, direct case relation matching involves checking that the case

relation names are the same and that destination instances represent two instances being matched. In the query and caption graphs, this corresponds to checking for the existence of edges between the instance vertices. For example, in Figure 5.2, we are attempting to match the *theme* edge emanating from the "noun(query1-1)" instance vertex with a *theme* edge emanating from the "noun(5824-2-1)" instance vertex where the destination vertices are related by the "missile" class hierarchy. In this step, we are only concerned with matching direct edges from vertices.

For other *cases*, we have a single-term quantity or quality (e.g., "2" or "red") that is to be compared. These values do not represent edges but are values describing an instance vertex. These values can be found in the *quantity*, *state*, and sometimes *attribute* relations. Slot values that contain quantifier or qualifier modifiers are matched exactly. However, these values can be either a predicate or atom.

Example 5.9. Matching qualifiers and quantifiers.

(a) *Query* = blue missile

```
inst(noun(query-1-1), missile).
attribute(noun(query-1-1), blue).
```

Caption = blue Harm missile

```
inst(noun(123456-1-1), Harm).
attribute(noun(123456-1-1), blue).
```

(b) *Query* = missiles

```
inst(noun(query-1-1), missile).
quantity(noun(query-1-1), plural).
```

Caption = two Harm missiles

```
inst(noun(123456-1-1), Harm).
quantity(noun(123456-1-1), plural(2)).
```

In Eg. 5.9(a), the *attribute* relations do not represent edges; the values are qualities. We easily see that "Harm" and "missile" match (score of 1) and both "blues" match (score of 1), hence we assign a total score of 2.0 to this match. In the *attribute* records for Eg. 5.9(b), "plural" is handled as a predicate name in the caption whereas it is a value in the query. We handle "plural" in the query as a predicate name as well, but with no arguments. This treatment allows both predicate names to match. However, since the values do not match (i.e., "2" cannot match with anything in the query), we are done. To compute the score, we handle both "plural" terms as consisting of two components where the predicate name (i.e., "plural") has a weight of 0.5 and the numeric value has the other weight of 0.5. "Harm" and "missile" again match (score of 1), both "plural" predicate names match (score of 0.5), but the predicate values do not. Hence, the total match score is 1.5.

Situations where the second argument of an *attribute* record contains references to other instances can be found in Appendix F. They are handled just like any other case relation involving edges. We estimate that direct case relation matching takes 35% of the total fine-grain match time.

3. Indirect Case Relation Matching (Phase 1)

The previous step examined just the endpoints of a query edge to see if a correspondence could be found in the caption. However, situations exist which fall outside of this type of checking which require us to follow some inference chain or path in the graph. We must now look at the slot values for each query instance to see what values were not matched.

We will use Eg. 5.7 and Figure 5.1 to demonstrate the basic problem. In Figure 5.1, we have the notion that there is a missile situated on an F/A-18 aircraft. The *location* record in the query emanates from the verb vertex "assemble" whereas in the caption, it emanates from the noun vertex "AIM-9M." To effect this match, we examine the *theme* case for the "assemble" and find the "missile" instance. We then discover the match between "missile" and "AIM-9M." From this we are able to infer that the location of the "missile" and "Sidewinder" is on an "F/A-18" (specifically, F/A-18A). There are three possible score assignments we can make for this type of match: 1 (*location* only), 2 (*location* & *theme*), or 3 (*location*, *theme*, & *activity*). If the caption had read "Sidewinder missile mounted on an aircraft," then we could easily say that we have an exact match with $m = n_q$ (= 5). If so, we then surmise that we would prefer the exact match over one requiring an indirect case relation match. Otherwise, why mention "mounted" at all? Alternatively, we could have a situation where the missile is just on the aircraft, but maybe not mounted. On the other hand, the query and caption appear to be almost semantically the same; thus maybe they should be treated the same. For the present, we leave this as an open issue and assign a score of one.

From Eg. 5.7, another way that we can describe this step is by matching query verb-related cases against caption noun-related cases. In the example, we were attempting to match a verb *location* case record with a noun *location* case record. This match involved adding a specialized rule for locations; other situations can be handled similarly. We estimate that this step takes about 20% of the fine-grain matching given that we have specialized rules just for *location*.

4. Indirect Case Relation Matching (Phase 2)

One situation that this last step handles is the reverse of the previous step, i.e., we are attempting to match query noun-related cases with caption verb-related cases. For example, suppose the query read "missile on F/A-18" and the caption was "Sidewinder AIM-9M mounted on F/A-18A." In this situation we have $m = n_q$ (= 3). We can simply argue that we have no further records to match and since we have exhausted all of the query records, we can do no better.

Another situation that this step handles involves correlations. In Eg. 5.10, we have a case where two locations are explicitly stated as being related. This relationship is then expressed in the *location* record in the query. The caption, however, states no such relationship.

Example 5.10. Matching involving correlations.

```
Query  = Hot Line at NAF

    inst(noun(q-1-2),Armitage Field).
    location(noun(q-1-1),at(noun(q-1-2))).
    inst(noun(q-1-1),Hot Line).

Caption  = NAF airfield. overview of Cold Line and Hot Line.

    theme('noun(66695-2-1)',obj('noun(66695-2-2)')).
    theme('noun(66695-2-1)',obj('noun(66695-2-3)')).
    inst('noun(66695-2-1)',overview).
    inst('noun(66695-2-3)','Hot Line').
    inst('noun(66695-2-2)','Cold Line').
    inst('noun(66695-1-2)','Armitage Field').
```

"NAF" (Armitage Field) is the airfield at the Center. Hot Line and Cold Line are specific locations (parts) of the airfield. The instances in both the query and caption easily match and result in a match score total of 2. The *location* record in the query is the only thing that hasn't matched. However, we can improve the score by analyzing the *correlation* slot in the type hierarchy for "Hot Line" shown in Figure 5.3. Using the *location* record together with the slot information from the type hierarchy and examining the "Hot Line" instances in both the query and caption, we are able to infer that "Hot Line" in the caption is located at "Armitage Field" by virtue of the *part_of* correlation (plus the fact that both "Hot Line" and "Armitage Field" are locations). This inference allows us to match the *location* record in the query.

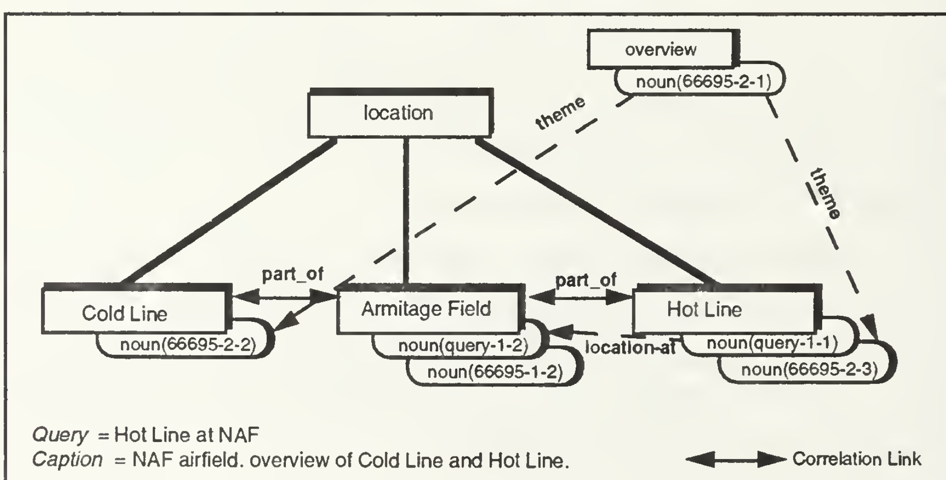

Figure 5.3. Matching using Correlation Link from Eg. 5.10.

A similar problem is listed in Eg. 5.11 and shown in Figure 5.4. Here we must follow two *location* edges before we pick up the *part_of* correlation link. This type of traversal can become quite complicated as we discover additional ways in which a query and caption can match.

Example 5.11. Matching involving correlations.

```
Query  = missile under aircraft

    inst(noun(query-1-2),aircraft).
    location(noun(query-1-1),under(noun(query-1-2))).
    inst(noun(query-1-1),missile).

Caption  = Bat missile sitting on ground beneath wing.

    inst('noun(110169-1-1)','Bat').
    theme('prespart(110169-4-1)',obj('noun(110169-1-1)')).
    location('prespart(110169-4-1)',on('noun(110169-4-3)')).
    activity('prespart(110169-4-1)',inhabit).
    location('noun(110169-4-3)',under('noun(110169-3-2)')).
    inst('noun(110169-4-3)',land).
    inst('noun(110169-3-2)',wing).
```

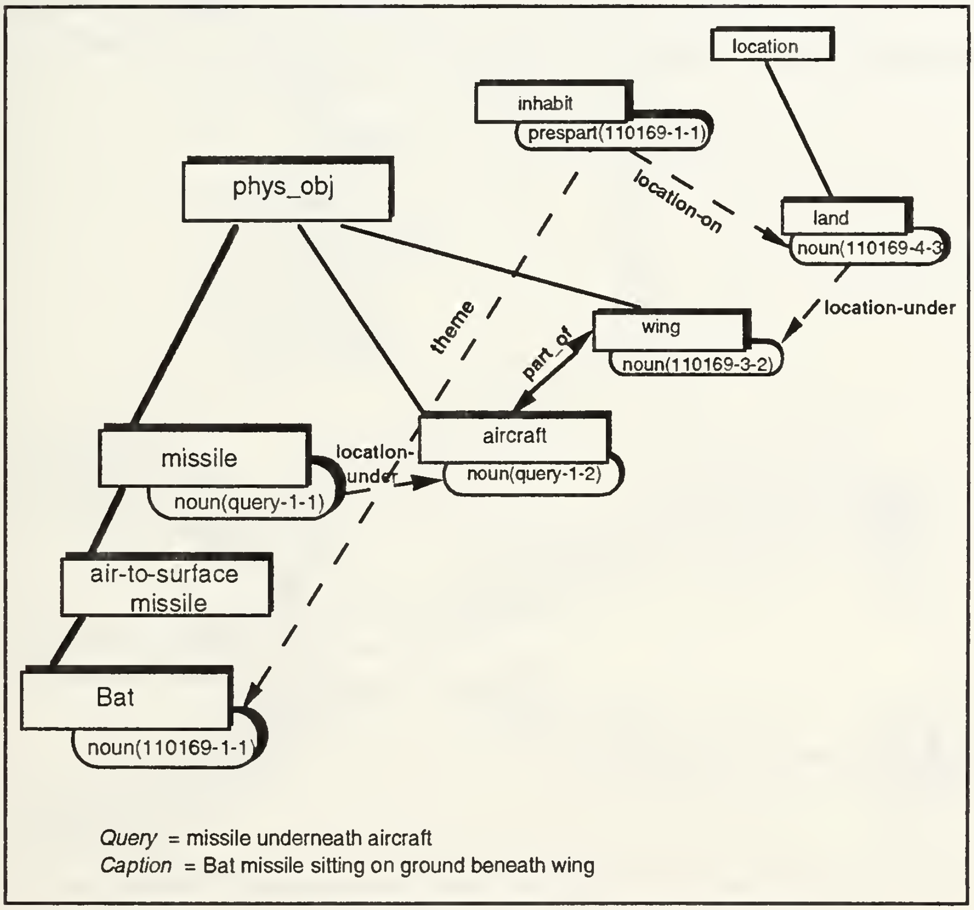

Figure 5.4. Matching using Correlation Link from Eg. 5.11.

One final example is the query "Naces seat ejection" with caption "Naces T-45 Goshawk seat ejection" (Caption 251272) where seat is *part_of* the aircraft.

5. Match Results

The fine-grain match result is a list of pairs, consisting of caption identifiers and match scores, m. We decided that the coarse-grain search threshold, k, and the maximum possible match score, n_q, should be included as part of the results in order for the user to determine how well/poorly the match was performed. Knowing how close to the minimum or maximum score a caption score is might provide or inhibit incentive for restating the query.

E. SUMMARY

Keyword matching based on class/subclass concepts allows a user to not have to remember every specific keyword. The user can use specific terms when they are known or general terms when they are either unknown or the user wishes to explore a wider range of concepts. Subgraph isomorphism for the case of directed graphs and undirected graphs are both NP-Complete problems [Garey 79]. Our matching approach has been assisted by the use of history lists to avoid unnecessary repetitions when exploring new match possibilities. The next chapter discusses match results based on the Photo Lab's and our queries.

VI. IMPLEMENTATION & PERFORMANCE

Chapter III provided an overall conceptual view of the system. In this chapter, we describe how the various pieces are implemented and some performance measures.

A. INTRODUCTION

MARIE was built using a client-server model and is written mostly in Quintus Prolog, version 3.1.1. The system currently runs on Sun Sparcstations under SunOS 4.1.1. The modules making up this model consist of MarieSearch (user interface for searching), MarieKeys (user interface for semantic database creation), MarieNLP (NL interface, semantic database creation, and coarse-grain match), and MarieFine (fine-grain match). The approximate number of lines of code (Prolog + C) is shown in Table 6.1. Communication is handled by TCP/IP sockets provided in the Quintus Prolog TCP library. The overall communications flow is shown in Figure 6.1. We have tested the system with 217 captions and 65 test queries. The remainder of this chapter describes each of these four modules in detail.

TABLE 6.1. MODULE SIZES.

Module	Lines of Code
MarieFine	412
MarieKeys	70
MarieNLP (DBG)	10895
MarieNLP (Ours)	7004
MarieSearch	553
Library (DBG)	546
Library (Ours)	1114

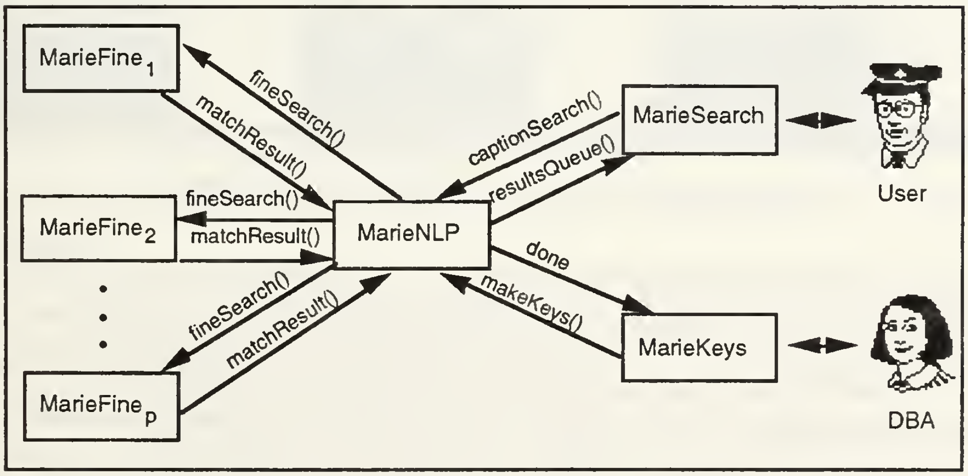

Figure 6.1. MARIE Client-Server Communication.

B. MARIESEARCH

MarieSearch is the user interface module for querying the database. It was developed using ProWindows 2.0 for X-Window environments. The module consists of three windows: a window for entering a query and listing the search results, a second window to view the registration data, and a third window to view or listen to the multimedia data (see Figure 6.2). The user interface was organized as a separate process to allow different user interfaces to be plugged into the system with minimal change. One such example is described in Guglielmo et al. (1992).

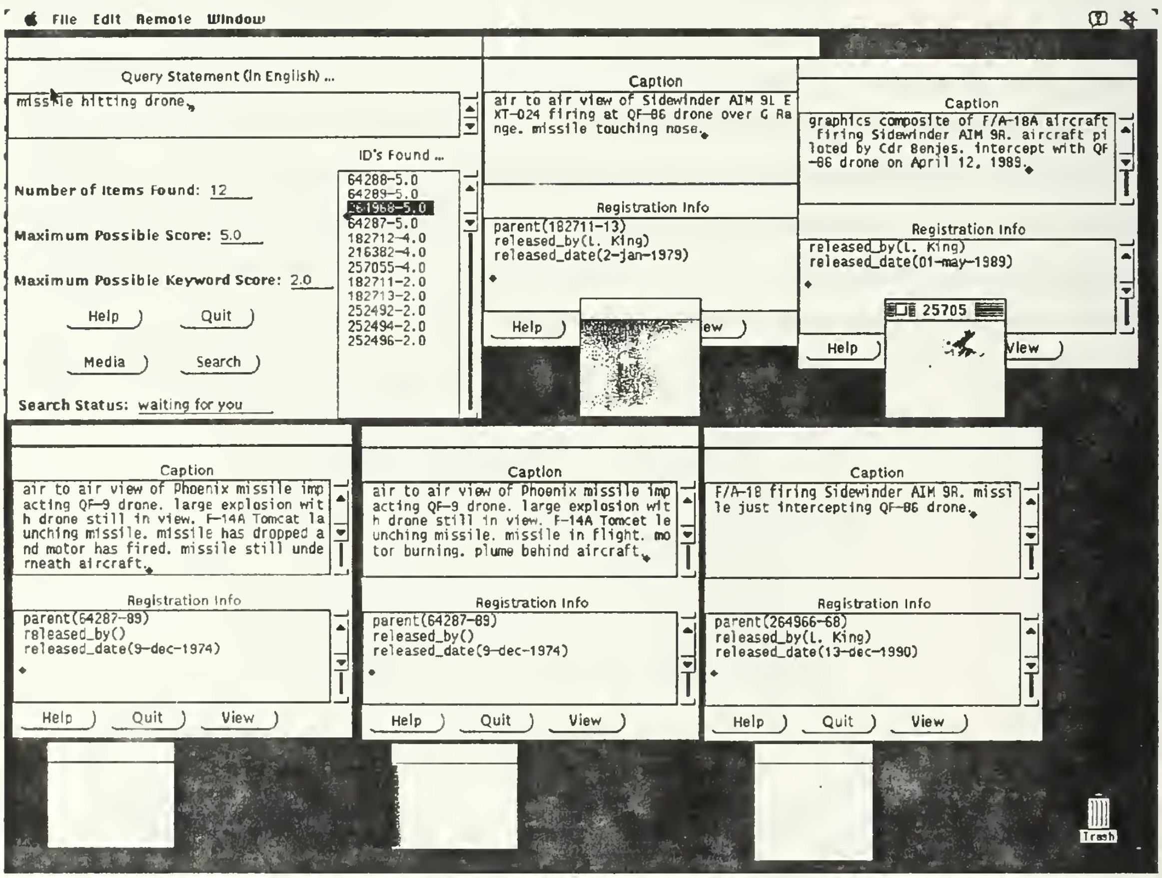

Figure 6.2. MarieSearch Window Environment.

As part of the query process, the user must select the multimedia type: graphic, image, text, video, or voice; the default is image. MarieSearch then takes the type selection and the user query and constructs a *captionSearch*() message to be sent to MarieNLP. When MarieNLP is finished, MarieSearch receives the message *resultsQueue*() with a list of caption identifiers and caption scores satisfying the query. To aid the user in determining the effectiveness of the match, the coarse-grain match threshold, k, and the maximum

possible match score, n_G, are also returned and displayed to the user. If some error occurs such as an unknown word is encountered, MarieNLP sends the message *error*() with a description of the error.

C. MARIEKEYS

MarieKeys is used to build the keyword indices and logical form files in the semantic database. It reads a file consisting of caption file locations as to where the caption data can be found. (We currently have the caption records in individual files as well as in the relational database because of the way the system was incrementally built. We can easily alter MarieKeys in the future to read the caption data directly from the relational database.) Using this information, MarieKeys then reads in the caption data file and formulates a message to send to MarieNLP. This message, *makeKeys*(), instructs MarieNLP to enter the new caption into the semantic database. When MarieNLP is finished, it returns a *done*() message allowing MarieKeys to either send the next caption or stop if it has no more captions to enter.

D. MARIENLP

The MarieNLP module is the heart of the system. It accepts messages from either MarieSearch or MarieKeys containing either queries or captions respectively, generates the logical form, and either initiates the matching process or updates the semantic database. It also communicates with MarieFine processes to perform fine-grain matching on supplied captions. This section will describe the NL processing component, update of the semantic database, and the matching process (specifically the coarse-grain match).

1. Natural Language Processing

Recall from Chapter IV that the NL processing part of the system was adapted from the DBG Message Understanding System. A sample lexicon was included as part of the system to which we added our domain words and other enhancements. The type hierarchy, however, is new to our system.

a. Lexicon

The structure of the lexicon is discussed in detail in Chapter IV as well as our modifications to it. The lexicon currently has 1942 words. On the basis of on the morphological structure of each word, additional lexicon entries are created in the knowledge base to reflect plural forms, past tense, present tense, etc. The total number of words after this processing is 3832. The lexicon is loaded into the system at the start of MarieNLP.

The DBG system originally had a subsystem to allow a query user to add new words when an unknown word was encountered. We decided against using this for the following reasons. First, in entering a new word, there is some simple information that we can easily ask from a user and then enter it into the lexicon. However, the user must also know how the word is mapped to the type hierarchy and what different things can be said about the new word with respect to other words (in particular, correlations). An ordinary query user of the system will not know this information. Furthermore, if a user enters a new concept that is

unrelated to any concept in the database, then we know that there are no captions in the database already. Hence, there is no reason to prompt for new words unless we can relate it to an already existing word as a synonym.

b. Type Hierarchy

The type hierarchy was originally implemented using Elsa Software's LAP object-oriented Prolog programming environment, version 3.3.1. The environment was later dropped in the implementation for the following reasons. First, class names had to be unique. We preferred to store words that have both noun and verb representations under one class. However, to distinguish between the two, one of the names had to be altered. If not, the children of a word had to somehow be distinguished. In addition, we needed to distinguish between word senses within the noun category or verb (e.g., two different senses of "range"). The approach we have taken in our own implementation is to associate *sense* identifiers with each word. For example, the noun class "stand" is distinguished from the verb class "stand" in that the former has the sense identifier "noun-1" while the latter has the identifier "infin-1." The number in the sense identifiers allows us to distinguish between different word senses for the same word type.

Second, we needed to establish correlations between classes (e.g., "wing" is *part_of* an "aircraft"). However, a *named semantic link* in LAP could only be established between classes. Once we established a *part_of/has_part* link between two classes, we couldn't use the same link name for other classes. This forced us into a workaround -- creating one correlation slot inherited by all classes and specifying the correlations as slot values for each class. A related problem was the need to establish a semantic link between the verbal form of a word and its verb form (e.g., "elevation" and "elevate"). We ran into the same problem as with correlations.

Lastly, under LAP 3.3.1, the size of the LAP knowledge base was approximately one megabyte for 60 concepts. When we went to 140 concepts without adding additional slots or methods, the knowledge base size was four megabytes. This type of increase was an area of great concern knowing that we needed approximately 800+ classes. Analyzing the code we had written and expected to write, we saw that there were many features that we did not require in LAP. Hence, we conjectured that a simple home grown system would function equally well.

The type hierarchy is implemented separately from the lexicon, but there is no reason why they cannot be combined. We implemented them separately to avoid modifying the DBG lexicon building routines, which was a nontrivial task. However, the two structures are related by means of the *fp* and *fpcat* predicates in the lexicon. The *fp* predicate value will specify an ancestor concept for categorizing a word (nouns mostly) so that this information can be used to assist in creating the logical form; the *fp* value need not necessarily be the parent of a word. The *fpcat* predicate gives the canonical form for a word as it is defined in the type hierarchy. The hierarchy is defined by using *a_kind_of* and *slot* predicates. When

MarieNLP is started, an internal type hierarchy is built using a breadth-first approach, i.e., we create the node at level 0, then the nodes at level 1, the nodes at level 2, and so on. Nodes are assigned level numbers that facilitate the creation of an ordering scheme when we which to determine ancestors or descendants of a node. Both *child_of* and *parent_of* node records are used as Quintus Prolog hashes on the first argument (class name). There are presently 831 noun and verb concepts represented in the type hierarchy.

c. *Parser*

The parser implementation follows the same methodology as the original DBG system; i.e., everything is done in stages. Once the syntactic parse tree structure is produced, the functional parse rules are applied. These rules work on different parts of the tree and are applied from left-to-right (from the beginning of the sentence) and bottom-up (from the actual words) in the tree. The functional parse records are processed bottom-up from the list that is created -- creating class instances first, specializing and adding descriptors to the instances, handling relationships, and then performing transformations and inferences.

2. The Semantic Database

Before we can process a query, both the keyword and caption logical form files must be created or updated with new multimedia data captions. The semantic database files are organized hierarchically in a Unix filesystem (see Figure 6.3). The semantic database files are stored separately from the relational database files.

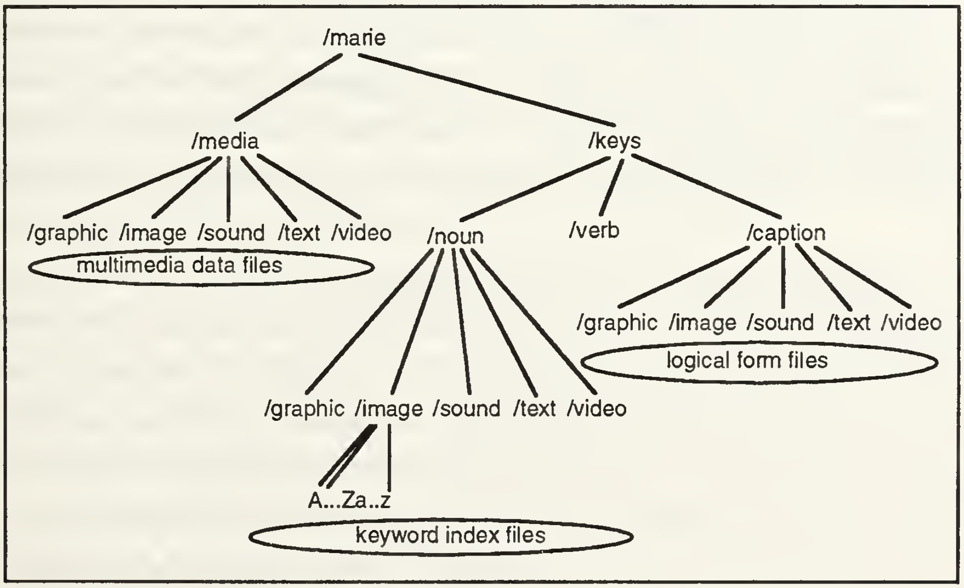

Figure 6.3. Semantic Database File Structure.

a. Keyword Index Files

Each keyword index file pertains to a class name and contains records of the form (*CaptionId*, *Case*). An index file is initialized by transcribing the keyword update records from a caption that has just been parsed to the file; we then assume we have a sorted file of one caption. New keyword update records are inserted into the index files in sorted order based on *CaptionId*. Let n be the number of records in an index file and m the number of records to be inserted into the same index file for a new caption. Insertion time is then $O(n + m)$. There are presently 468 noun index files consisting of 1597 records with an average of 3.412 records per file. For the verbs, there are 38 files, 154 records, and an average of 4.053 records per file.

Recall from Chapter V that keyword update records are used to modify the database and that the list of cases for a particular class in a record may not be in sorted order. Presently, keyword records need not be sorted on *case* as well because we are looking at a small number of records (< 4) for a caption and the overhead of invoking a sort outweighs the benefits. We are also not processing on the *case*.

b. Logical Form Files

As shown in Figure 6.3, the logical form files are stored separately from the keyword index files. Each file is identified by a caption identifier and contains its logical form records. There are 217 files holding 4153 records with 19.185 average records per file. The file contents can be found in Appendix F.

3. Coarse-Grain Match

a. Setting up the Queues

Recall that the first part of the coarse-grain match is an enhanced keyword lookup. On the basis of the nouns in the user query, we attempt to find all captions that contain the query nouns as well as their subclasses as defined in the type hierarchy. Two hashed priority queues (*CgsQueue* and *FgsQueue*) are used to hold those caption identifiers and scores that are below and at or above the coarse-grain match threshold.

Recall the discussion of the *union-count* algorithm in the previous chapter. The caption identifiers are read from each keyword file (and can be filtered against a user-supplied *case* in the future) and *union'd* into a temporary list for a particular query keyword. In addition, we also perform some semantic analysis in order to see if some of the information might be contained within the relational database. If so, the appropriate SQL command is executed and a second list of caption identifiers is created. Both lists are then *union'd* to remove the duplicate caption identifiers (remember that the same information might be found in the registration data as well as in the caption). The resulting list is then inserted into the *CgsQueue*, one caption identifier at a time. If a caption identifier already exists in the queue, then the identifier's reference count is incremented. If it is new, it is initialized to 1. When all files have been processed, the reference count for each caption identifier is thresholded against the coarse-grain match threshold. Those captions that meet or exceed the threshold are entered into the *FgsQueue* for fine-grain matching.

We present some general comments about the procedure. First, the treatment of a SQL call is being handled like a keyword search. For example, checking for a specific photograph location is one SQL call, checking for a photograph date is another, and so on. If we issued one SQL call containing several fields *AND'd* together, then a mechanism to combine the results (from the keyword search and SQL search) and update the threshold score correctly is nontrivial. We need to know when a caption identifier for a specific field was found in both and when it was not, so that we do not count a keyword twice. Second, insertion into the *CgsQueue* could be parallelized if we handled each keyword lookup and insertion into the *CgsQueue* as a separate task. However, the results of the SQL retrieval must be combined with the keyword match before insertion into the queue. Finally, one could conjecture that those caption identifiers that are eligible for movement into the *FgsQueue* could be moved asynchronously with the insertion of caption identifiers into the *CgsQueue*. However, this only occurs once the *k'th* keyword file is inserted into the queue.

The priority queues and supporting operations are written in 'C.' The relational database is implemented in the Oracle RDBMS, Version 6.0.33.1. ProDBI 1.0 is used to communicate between the RDBMS and the match routines.

b. Scheduling the Fine-Grain Match

In the early development phases, the coarse- and fine-grain match routines were contained within one process. We had hoped to have one common type hierarchy that could contain both the query and several caption logical forms with separate tasks operating on a query-caption pair in parallel. Quintus Prolog, however, did not provide for parallel processing of predicates within an executable. Closer examination of the match indicated that several fine-grain matches could be done in parallel (each with its own type hierarchy) without affecting one another. Hence, we separated out the fine-grain match module and implemented it as a separate process. The overall search module within MarieNLP that controlled the coarse-grain match then submitted match requests to the fine-grain match processes.

As part of the MarieNLP startup, the number of processors, p, to be used for fine-grain matching can be supplied as an argument (default is one). MarieNLP then connects to p sockets attached to p MarieFine processes that can all be contained on one workstation or spread out amongst p workstations. MarieNLP sends out a *fineMatch*() message containing the query logical form, caption identifier, and multimedia type. It receives back a *matchResult*() message indicating the caption identifier and score.

The scheduling algorithm always removes the first element from *FgsQueue* to formulate a match request message. It also favors *sends* over *receives*. Thus, MarieNLP will send out p match request messages before waiting for a response. When it receives a response, it will cache the results to memory and send out the next match request before waiting again. This cycle continues until the *FgsQueue* is empty after which time it just waits for responses. A counter is used to indicate the number of total match requests and is decremented each time a response is received. When the count reaches 0, there are no further messages

to be received and the matching is complete. The Quintus Prolog TCP library provides a time-out mechanism to abort a send or receive in the event a machine should crash.

Once all of the requests are received, the caption identifiers and scores are put into a list, sorted in decreasing order on the score value, and put into a *resultsQueue*() message to be sent to MarieSearch. This message also contains the coarse-grain match threshold and the maximum possible match score. Note that this process could also be done asynchronously. That is, a user need not wait until all of the captions are matched, but could receive one as soon as a MarieFine process returned a match result. This modification would require a message be sent to MarieSearch either before or after handling a new send, incurring some small delay. The problem is then in MarieSearch that has to handle random TCP messages and user inputs; this would require use of an interrupt handling mechanism.

c. Data Store

The majority of this data are presently stored on magnetic disk but can be moved to optical storage (presently a 10-platter magneto optical jukebox) when needed. The reason to move it to optical storage is that the data are primarily archival in nature and access to it is only made once a desired caption identifier has been chosen. The indices (relational and keyword) to these data are on magnetic disk.

To handle certain retrieval issues, we have decided to store the multimedia data in two qualities. First, low-quality for examination during query processing since it requires less storage and can be loaded and displayed quickly. Second, high-quality for detailed examination. For example, a high-quality 8" x 10" image scanned in at 400 dots per inch (dpi) using 32-bit color takes approximately 25+ megabytes (MBs) of storage. For a quick examination of the image, we can use less dpi and 8-bit color while also scaling the image to a smaller size. We have currently been using 75 dpi with 8-bit color and a scaling factor of 0.15. In GIF format, the resulting files range in size from 3 kilobytes (KBs) for black and white to 7 KBs for color. With this size it is also feasible to store some of these scaled-down images on magnetic disk permanently.

E. MARIEFINE

MarieFine performs the fine-grain matching of a query against a caption. Upon receiving a *fineSearch*() message from MarieNLP, it loads both the query (from the *fineSearch*() message) and the specified caption logical form from the magnetic disk file into the type hierarchy and performs the matching. When the match is complete, it sends a *matchResult*() message to MarieNLP together with the caption identifier and score. It then awaits the next match request. Note that the query logical form records need to be sent in every *fineSearch*() message because we never know when the user may have entered a new query.

The coarse-grain match is currently a serial process. The fine-grain match, however, operates in parallel by having *p* MarieFine processes executing. When we start a MarieFine process, we specify an identifier for that process (0, ..., *p* - 1). This identifer is used to designate the particular socket that the MarieFine process will use to communicate with MarieNLP. The MarieFine processes are in essence servers; they must be started before the MarieNLP process. The MarieFine processes can all execute on the same machine as the MarieNLP process, or on separate machines on the network to achieve concurrency.

F. CAPTION ALTERATIONS

The original captions encountered at the Center have been written by many people over many years. There was never any policy or guidelines provided for writing captions, and there was no one to provide any editing function with respect to grammatical constructs and spelling. Hence, many errors and inconsistencies arose. As we stated in Chapter III, we started with the automated retrieval system in use. In some cases, examination of handwritten individual captions located on the folder of each photograph has revealed grammatical constructs that can be more easily understood and parsed. The individual caption for photograph 262868 from Table 3.1 is shown in Eg. 6.1. The quality of the caption depended largely upon the photographer taking the picture. Rewriting of captions was done by a database administrator when the retrieval system was automated. Disk space and field length constrained how and what was specified in the caption used in the system. As a result, captions were summarized and, in some cases, poorly written and

Example 6.1. Caption 262868.

```
Sidewinder AIM 9R missile mounted on F/A-18C BU# 163284 aircraft, nose
110. Closeup side view of missile on outboard wing pylon.
```

punctuated to conserve space. The caption summaries, like the one in Table 3.1, are what we refer to as *supercaptions*. We have chosen to deal with individual captions initially because they provide an actual description of the scene in a particular photograph and they are easier to handle. We are using the existing captions contained in the Ingres database as much as possible and augmenting any with the particulars of the individual caption when supercaptions are encountered. The reason is to allow retrieval of a specific individual photograph or set of photographs depending on the generality of the query.

As explained in the previous paragraph, some of the captions were altered manually by us to clarify them. The following sections illustrate the problems and the types of alterations that we made for the captions (see Appendices A and B).

1. Case Sensitivity

Captions stored within the Ingres database and the individual photograph folders were all written using one case -- upper case. The first decision we made was to allow both upper and lower case characters

and use the upper case characters for acronyms and proper nouns. This alteration had to do more with the way a noun is conceptualized as opposed to computer processing.

2. Appositives

Eg 6.2(a) presented the first case of dealing with appositives. In this case, the appositive is describing the abbreviation and is contained within the noun phrase. We examined the appositives in some captions and they were listed within the noun phrase with no delimiters. Other captions (see Eg. 6.2(b)) delimited them with commas. What we observed was an inconsistent method of delimiting the appositives. To handle the inconsistency, we specified the appositive by enclosing it in parentheses and removed any extraneous punctuation characters. For those cases where the appositive defined the abbreviation, we listed the full name in the lexicon. Hence, we included an entry for 'Advanced Technology Stealth Bomber' with a semantic representation of 'ATB' in addition to defining 'ATB' outright.

Example 6.2. Original captions containing appositives.

(a) Caption 900304.

```
Air to air, Northrop ATB 'Advanced Technology {Stealth} Bomber', B-2
aircraft, first public flight late 1989.
```

(b) Caption 10862.

```
Sparrow III, Guardian of the Skies, operational with Seventh Fleet in
Western Pacific. Four missiles on underside of F3H-1 BU# 137010, Point Mugu
7010 on tail. Air to air view from side during firing of one missile.
```

3. Commas

A related problem dealt with the inconsistent use of commas. In Eg. 6.2(a) there are really two sentences or phrases; the first phrase terminates after the word "aircraft." To show this, we replaced the comma with a period. The other comma after "air to air" was not obvious until we discovered the meaning of "air to air" - "air to air view." Closer examination of other captions with commas similarly placed after verbal type nouns led to the realization that the comma really meant "of." Hence we replaced this comma with the preposition "of."

In a different twist, Eg. 6.3(a) shows the comma preceding the last noun phrase in the sentence; "nuclear program" refers to the guided missile submarine SSG-N1. In Eg. 6.3(b), "front view" refers to the Sidewinder missile on the stand. Syntactically we have a difficult time differentiating between the two. In one case, the word refers to the immediately previous noun, in the other case it does not. Both preceding noun phrases are contained within prepositional phrases, so we cannot use the prepositions to help us. The fact that "view" and "program" are both verbal nouns (i.e., have verb forms as well as noun forms) does not help either.

We handled these problems as follows. Since "nuclear program" in the first sentence modifies the preceding noun, we enclosed it in parenthesis and made it an appositive. We make the general rule that an

Example 6.3. Original captions with commas preceding the last noun phrase.

(a) Second sentence of Caption 10880.

```
Artist conception of FBM guided missile submarine SSG-N1, nuclear program.
```

(b) Caption 262866.

```
Sidewinder AIM 9R missile on stand, front view.
```

appositive modifies the preceding noun. In Eg. 6.3(b), we could have dealt with "front view" as an appositive as well by placing it after "missile." We chose instead to move it to the beginning of the sentence and follow it with the preposition "of" making it consistent with other captions and their use of view.

In Eg. 6.4(a), "front view" refers to the pedestal structure. Unlike Eg. 6.3(b), we cannot move it to the beginning of the sentence, otherwise "front view" would see "load test" as the noun it was referring to. In this case, our only choice is to make "front view" an appositive to the pedestal structure as shown in Eg. 6.4(b). Our goal was to keep the modifications simple and not have to rewrite the entire caption.

Example 6.4. Captions with commas preceding the last noun phrase.

(a) Original Caption 188716.

```
Load test on XN-01 pedestal structure with 45 degree elevation, front view.
```

(b) Modified Caption 188716.

```
Load test on XN-01 pedestal structure (front view) with 45 degree
elevation.
```

4. Missing Prepositions

As we saw in caption Eg. 6.2(a), the comma represents a missing "of" preposition. "1989" in Eg. 6.2(a) also has a missing preposition. As it reads, "1989" is the head noun of the noun phrase preceded by adjectives and a noun acting as an adjective. To obtain a correct interpretation of the noun phrase, the preposition "in" needs to be included before it.

Another situation was found in the first two phrases of Eg. 6.5. that contained a comma problem as well as two independent sentences. What the caption writer really should have said was the "facilities at Harvey Field." The "view of ..." was a second thought or sentence. We made these adjustments to improve readability and parsing.

Example 6.5. Caption 124 with comma connecting two independent phrases.

```
Facilities, Harvey Field, view of runway with aircraft from inside tower.
```

5. Missing Nouns

In the first sentence of Eg. 6.6(a), we have one case where a noun, in particular "aircraft," is missing after "151562." This might not seem important except for the fact that "BU#" becomes the head noun of the noun phrase. This means that "A-6A," "nose #562," and "NG on tail" are all referring to "BU#" when in actuality, they refer to the missing noun "aircraft." In over 95% of the captions, we have a noun following an identifier like "BU#." The second sentence of Caption 10862 shows the same problem. Our solution was to insert the word "aircraft" after "151562" and "137010."

Example 6.6. Captions with missing head nouns.

(a) Caption 110169.

```
Bat and Standard Arm missiles with A-6A BU# 151562, nose #562, NG on tail.
Standard Arm mounted on wing with Bat missile sitting on ground beneath.
```

(b) Second sentence of Caption 10862.

```
Four missiles on underside of F3H-1 BU# 137010, Point Mugu 7010 on tail.
```

A similar problem appears in the latter half of the second sentence of Eg. 6.6(a). At the end of the sentence, we do not know whether "beneath" is referring to the "wing" or the "Standard Arm" missile. We could have chosen to deal with the adverb form of the word or the prepositional form that required a noun at the end of the sentence. We chose the latter solution and inserted a noun (wing) at the end of the sentence to make the sentence less ambiguous.

A variation of this last problem concerns missing nouns that function as themes to verbals. Examine the underlined occurrence of "view" in Eg. 6.7. We are hard pressed to find out exactly what "view" is referring to, i.e., it could be an excellent view of the F/A-18A, the Harm missile, the pod, or MK 91 bomb. If we have such difficulty, our program would have just as much if not more. Therefore, we found it necessary to explicitly state what the reference to the view really was.

Example 6.7. Caption 231373 with unknown theme.

```
Air to air view of Harm missile launched from F/A-18A BU# 161720 aircraft
(nose 102) over NWC ranges. Harm, pod, and MK 91 Silver Bullet uploaded.
Excellent view just before firing.
```

This problem is not true of all captions. Examine "view" in Eg. 6.8. In this case, we are almost certain "view" refers to the ACIMD missile. Our program should be able to infer this as well. Hence, we do not need to explicitly state the reference for this caption.

Example 6.8. Caption 228544 with inferable theme.

```
ACIMD (Advanced Common Intercept Missile Demonstration) missile on stand in
front of Mich Lab. Full side view with Kennedy plaque on rock at rear.
```

6. Noun Ordering

In complex noun phrases, the task of determining the relationships is difficult enough without complicating the problem by having the nouns out of order. In Eg. 6.9(a), the head noun (last noun) is the US Navy and all the previous nouns are somehow modifying it. Examining the phrase more closely, "helicopter" should be the head noun. Hence, we move it to the last position to reflect that fact. "USN" on the other head is an organizational unit and such units are usually found at the beginning of the phrase. The remaining terms in this case do not make any difference except that "BU#" should precede and be adjacent to "149033." The rewritten caption is shown in Eg. 6.9(b).

Example 6.9. Captions with missing head nouns.

(a) Caption 161045 with nouns out of order.

```
BQM control helicopter UH-2A BU# 149033 USN
```

(b) Caption 161045 with nouns corrected.

```
USN UH-2A BQM control BU# 149033 helicopter
```

7. Registration Data within Caption

In Section A, we saw a registration record from the *visual* relation for identifying photos. Examination of the captions revealed that some of the data in the registration data fields were also contained within the caption, most notably, the date the picture was taken (*Date-Orig*), *Location*, and *Photographer*. Since retrieval entails both semantic and SQL (on relation attributes), we have removed redundant information from the captions that explicitly appears in the registration record fields. However, additional location information that is not contained within the *Location* field has been retained in the caption. An example is shown in Table 6.2. Notice that the date, location, and photographer appear in both the *Caption* and the *Date-Orig*, *Photographer*, and *Location* fields respectively. It is easier and more efficient to perform SQL queries on these later three fields than on the caption field.

In addition, upon examining this caption and that in Table 3.1, there is also other definable registration information that is not defined explicitly in the relation at present such as: location of the

original photograph, the date the photograph was released to the public, and the person authorizing the release. Such information appears on the individual photograph folders, but not in the relation. We have also taken this information out of the captions with the understanding that it would be more suitable to place it in predefined fields.

TABLE 6.2. REGISTRATION DATA FOR CAPTIONS 218178-89.

ATTRIBUTE	VALUE
Designator	LHL
Id	218178-89
Quantity	12
Date-Orig	29-jul-1967
Retention	H
Medium-Info	4X5 BW
Photographer	J. PARTIN
Customer	C. BLANCHARD
Code	327
Location	USS FORRESTAL
Date-Loaded	15-sep-1981
Caption	USS FORRESTAL, CVA-59. FIRE ACCIDENT, ACTION VIEWS. USS REPERTUS DD-851 AND HELICOPTER STANDING BY TO ASSIST CREWMEN FIGHTING FIRE, EXAMINING BURNED REMAINS. CAPT JOHN D. BELING INSPECTING DAMAGE TO FLIGHT DECK. ORIGINAL AT NPC,RELEASED BY NPC. PHOTO'S TAKEN IN GULF OF TONKIN ON 07/29/67. STUDIO COPIES BY J. PARTIN.
Class	U
Cross-Ref	

8. Explicit versus Implicit use of "View"

In examining the captions, we came across situations where the phrases "view of," "view showing," "front view of," etc., are used. Some captions do not explicitly contain the word "view" while others do. We make the assumption when dealing with image data that the caption describes some sort of view, hence view is implicitly understood and we should not have to explicitly state it. However, if it is a view from a certain aspect, like "front view" or "air to air view," then we retain this information. Also, if the caption is stated general enough to apply to several images, but a photograph contains a view of just a certain aspect, then we again retain the view. In Eg. 6.10, we would expect that the image has in it the image of the missile impacting the QF-86. The F-15 may not be part of the photograph. In the individual captions for Table 6.2, most of the second sentences began with "view showing ...". This use of "view" is not necessary because the rest of the sentence contained specific enough information to identify the event in the photograph.

Example 6.10 Caption 216382 showing retention of "view."

```
F-15 aircraft firing a Sidewinder AIM 9M missile against QF-86 target.
View of QF-86 and impact.
```

G. PERFORMANCE MEASURES

The NAWCWPNS China Lake Photo Lab image database contains historical photographs and slides of aircraft and weapon projects from the last 50 years. The photographs show mostly aircraft and missiles in flight, weapons systems configured on various aircraft, targets and drones being hit, aerial views of the scenery surrounding the Center, test equipment, etc. The photographs are used for reference, project documentation, and publications. Registration data is used to capture customer information about a particular photograph shoot such as to identify customer name and code, to note the date, time, and location of the shoot, to identify the photographer and describe the film used, and to attach cross references to related photographs. The caption describes the actual equipment and events in the photograph. The actual photographs are stored as either negatives or positives (slides); they are presently not stored in any digital form on disk. The database is used mostly by the current VISUAL retrieval system that uses keyphrases as the semantic index. SQL queries can be applied against both the registration data and the keyphrase records. A customer seeking a photograph of some particular event or equipment must provide the Photo Lab retrieval specialist with the specifics. The retrieval analyst is the only person allowed to search the database as it holds both classified and unclassified information.

The MARIE system was tested using a subset of this database. We selected 217 captions (and associated images) from the Center's 100K+ images encompassing 413 distinct sentences and noun phrases. We then asked the Photo Lab to formulate various queries based on the 217 captions. They came up with 46 queries and we added 19 queries of our own. From the 46 Photo Lab queries, 16 could not be answered using the present keyphrase approach. Two of the 46 could not be done by MARIE because they required either extra fields or an additional table in the relational database; another query required some data from one of the extra fields in conjunction with the caption data. We used both sets of queries to evaluate the system. Appendix G contains a list of these queries. To measure performance, we used the criteria suggested by several IR researchers -- among them, Cohen and Kjeldsen (1987), Stanfill and Kahle (1986), and Mauldin (1991). The criterion consists of *recall*, *precision*, *ease of use*, *response time*, and *maintenance*. We describe the system performance for each of these categories in the following sections.

1. Recall

From Cohen and Kjeldsen's definition, we define recall as shown in (1). Individual performance measures for each query for both the standard keyphrase approach and MARIE are shown in Appendix G.

$$\text{recall} = \frac{\text{number of captions judged good by MARIE \& good by Photo Lab}}{\text{number of captions judged good by Photo Lab}} \tag{1}$$

Average recall for the keyphrase approach for the 30 queries that could be done from the original 46 was 87.5%. Average recall for MARIE for 44 out of the 46 queries was 93.6 %. Average recall based on all 63

of the test queries for MARIE was approximately 91.9%. We should note that the reason recall is high for the keyphrase approach is because the retrieval specialist formulated the keyphrase query and knew what to use and include. An ordinary or naive user may not have such insight, which could result in lower recall.

2. Precision

To define precision, we need first define *fallout*. Again, based on Cohen and Kjeldsen's definition, fallout and precision are defined in (2) and (3). Average precision for the keyphrase approach was 95.2 %. Average precision for MARIE for the Photo Lab queries was 96.4 % and 94.7% for all of the queries. Precision for the keyphrase apprach is also influenced by the retrieval specialist formulating the queries.

$$\text{fallout} = \frac{\text{number of captions judged good by MARIE \& bad by Photo Lab}}{\text{number of captions judged good by MARIE}} \qquad (2)$$

$$\text{precision} = 1.0 - \text{fallout} \qquad (3)$$

Stanfill and Kahle remarked on the performance of keyword systems as follows: "If the searcher looks for any of several words (a disjunctive query), recall improves but precision goes down; on the other hand, if the searcher looks for documents containing all of several words (a conjunctive query), precision improves but recall suffers." This statement is true if search involves just the words present in the query. Use of a class/subclass hierarchy for categorizing related objects and fine-grain matching has shown this to be false. However, we should also note that the real-world queries we encountered were relatively simple.

3. Ease of Use

Ease of use for the naive user was of paramount importance in designing the system. It was envisioned that such a system would be used not only by the Photo Lab, but also by any Center personnel. Hence, the X-Windows environment was chosen because it provides a standard windowing environment for all UNIX-based workstations, Apple MACINTOSHES, PC's, and PC-clones. With respect to formulating a retrieval request, the user need only enter a NL statement, select the type of multimedia data to be retrieved, and initiate the search. As we stated earlier, the user receives a list of caption identifiers and match scores. The user can also look at the maximum possible keyword score and maximum possible match score to judge how effective the retrieval was. The Photo Lab is satisfied with the system at present, although it is not being used on a regular basis at this time. However, we feel one of the shortcomings, at present is not being able to analyze the query statement first and suggest improvements, such as in the way the query is phrased, emphasizing a greater use of verbs instead of numerous prepositions, elimination of redundancy (e.g., using just "AIM-9R" instead of "AIM-9R missile"), etc. Also, we have no user model with which to record a user's interaction with the system for further analysis as has been suggested by fellow researchers.

4. Response Time

Response time is the duration from the moment the user initiates the search until the moment results are produced. Various tests were conducted to evaluate the performance of this architecture. Sparcstation 2s and a Sparcstation 1+ were used to test the performance of MARIE. In Figures 6.4 and 6.5, we set up an environment where the MarieSearch and MarieNLP processes were executing on one Sparcstation 2, the file server was on a Sparcstation 1+, and up to five other Sparcstation 2's were used for MarieFine processes. Three different queries were used. The values of C indicate the number of captions that are being fine matched for a particular query. Note that the five processors all access one file server to obtain the database files, which is a bottleneck. Also, notice that the case of $C = 71$ (which has a large number of captions to match) falls in between the case of $C = 26$ and $C = 4$. We hypothesize that this happens because we are I/O bound on the file server disk in waiting to read the logical form from disk.

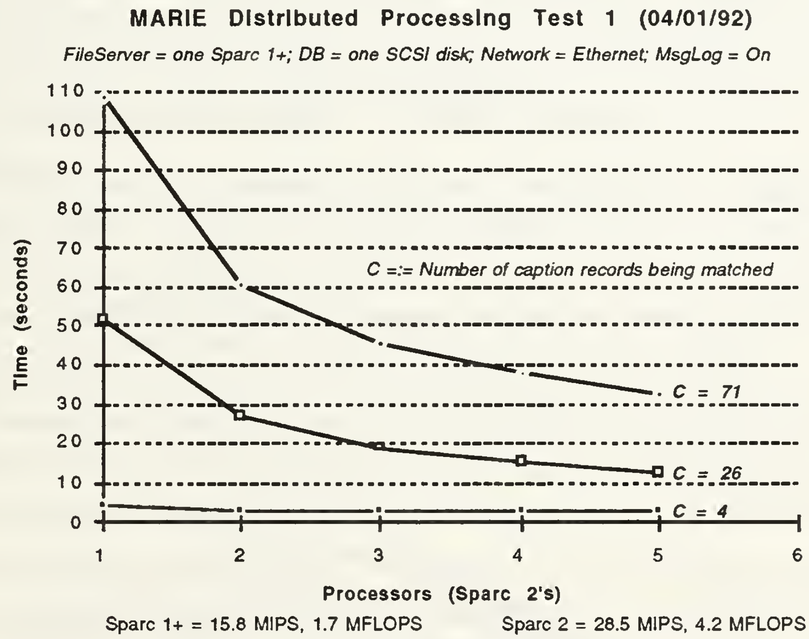

Figure 6.4. Fine-Grain Matching (Time vs. Processors).

Figure 6.6 shows the situation where we replaced the slower file server (Sparcstation 1+) with a faster file server (Sparcstation 2). Performance measures for up to 3 processors is slightly better. When we introduce the fourth processor (Sparcstation 1+) which is the slower workstation, performance worsens. Tests using multiple file servers were also conducted but were inconclusive as we did not have enough fine-grain match processors to stress the file servers.

The figures show the obvious. We can improve response time by using multiple processors to solve the problem. Issues of concern when using multiple workstations are first, distributing the logical

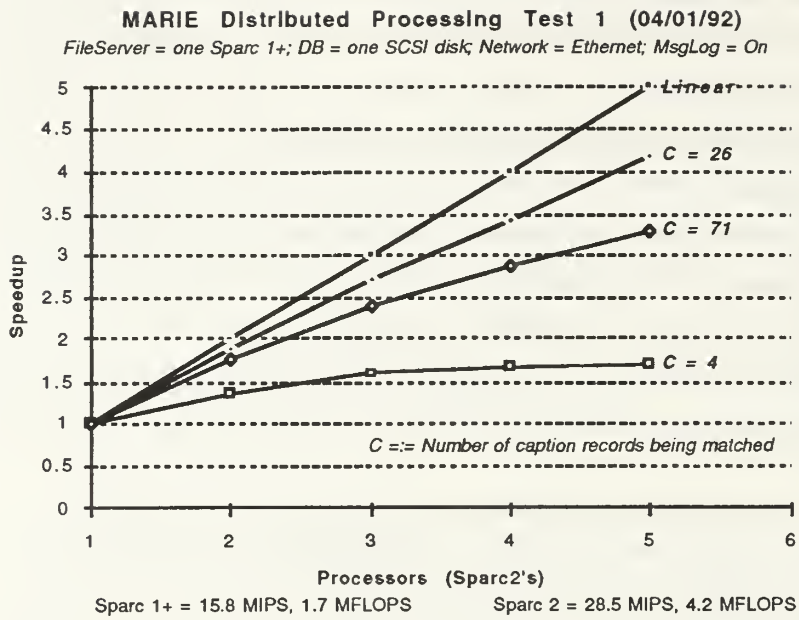

Figure 6.5. Fine-Grain Matching (Speedup vs. Processors).

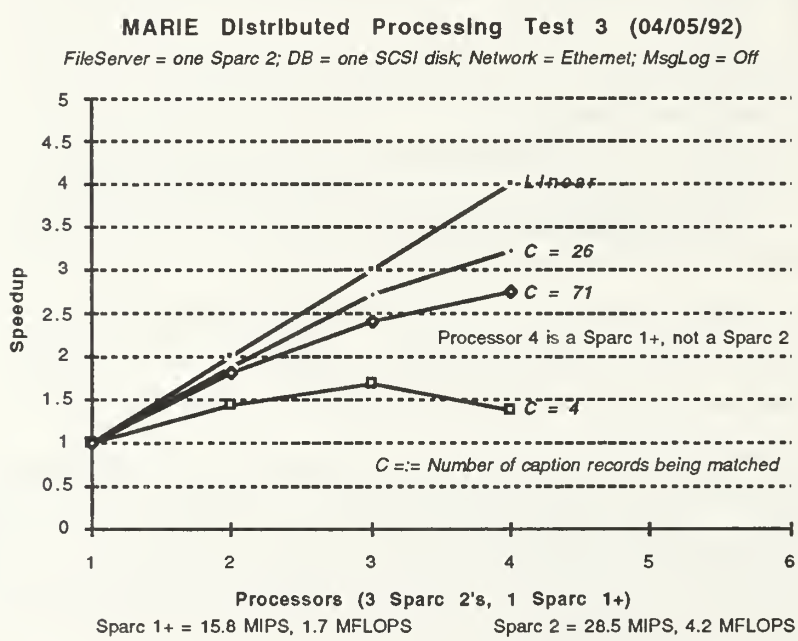

Figure 6.6. Fine-Grain Matching (Speedup vs. Processors).

form records across a number of servers to minimize the I/O bottleneck through one machine's disk; second, the amount of traffic on the local area network connecting the workstations during regular work hours; and third, determining the point of diminishing return for adding MarieFine processors to the problem.

Table 6.3 collects the response time measures for the Photo Lab queries in Appendix G, Section 3. The response time for a query is the amount of time it takes the NLP subsystem to produce the logical form (i.e., the %PARSE_STATS time in each of the test queries) and the search/match times (i.e., %SEARCH_STATS time). Column 2 indicates the number of words in the caption. Column 3 indicates the number of captions that were found for the query. Column 4 shows the query parsing time in seconds (real-time). Column 5 indicates the real-time seconds to perform the coarse-grain match. Column 6 indicates the real-time seconds MarieFine took to run where MarieSearch, MarieNLP, and one MarieFine process were all executing on the same machine (Sparcstation 2). Columns 7 through 11 show the real-time seconds for the MarieFine processes where MarieSearch and MarieNLP were executing on one Sparcstation 2, and the MarieFine processes were executing on one to five different Sparcstation 2s. We should note that the database was stored on a separate Sparcstation and accessed through the Network File System (NFS) from the other six Sparcstations. The last three rows of the table show the mean, median, and standard deviation for the queries.

Figure 6.7 presents a log-log chart based on the number of words in the query and the total time to process the query (parse + coarse-grain + fine-grain times) for the case of three MarieFine processes. Figure 6.8 shows a similar type chart based on the number of query results and the total processing time.

5. Maintenance

Maintenance, as defined by Mauldin (1991), is the amount of human labor required to keep the system operational. We have attempted to design the system so that all the retrieval specialist need concern himself with is the lexicon and type hierarchy. As we indicated earlier, both of these structures can be combined into one and thus require just one file to modify. However, deciding how to define a word is the more difficult of the tasks. At present, we have no performance data on entering new words versus entering new keyphrases.

The retrieval analyst must also ensure that whatever captions are written can be parsed correctly. The only way to indicate that they are parsed correctly is for MARIE to rephrase the caption from the parsed logical form records. Recall that the NLP subsystem will only fail if it encounters a word that is not in the lexicon. Hence, logical form records will always be produced and we should be able to write a NL generation program that can produce NL output from the records to validate the input. Such a capability has been studied by Verlardi et al. (1988) and those researchers building question-answering systems. The difficulty of this task rests on the quality of the NL output desired.

TABLE 6.3. PARSE + MATCH RESPONSE TIME MEASURES.

Query	Qword	Result	Parse	Coarse	Fine 1*	Fine 1	Fine 2	Fine 3	Fine 4	Fine 5
1.2	4	1	2.300	3.769	1.763	1.495	1.581	1.653	1.349	1.798
1.4	5	4	2.500	2.661	7.058	4.037	2.203	2.254	1.342	1.357
1.5	4	1	2.433	3.734	1.009	0.774	0.900	1.068	0.727	0.659
1.6	9	2	5.933	24.46	52.118	26.286	14.952	15.264	15.739	15.133
1.7	1	1	0.850	0.403	0.265	0.311	0.379	0.263	0.142	0.116
1.8	5	1	4.750	8.547	3.872	2.705	2.688	2.504	2.697	2.136
1.9	3	2	1.833	1.216	2.426	1.950	1.298	1.187	0.890	1.123
1.10(a)	8	1	6.583	1.459	3.967	2.796	2.727	2.728	2.885	2.724
1.10(b)	2	27	1.367	0.418	59.014	30.210	15.905	10.677	8.050	7.467
1.11	5	1	6.850	1.573	2.243	2.015	2.073	2.180	2.111	2.049
1.12	3	1	4.150	1.846	1.988	1.630	1.566	1.940	1.920	1.760
1.13	2	40	2.250	3.976	130.529	66.081	33.718	23.230	17.975	14.166
1.14	7	7	4.717	11.407	30.924	16.785	9.333	6.827	6.334	6.464
1.15	3	6	3.017	4.119	28.051	14.895	8.158	5.912	5.850	5.645
1.16	7	2	7.017	3.322	12.376	6.761	3.544	3.327	3.577	3.561
1.17	5	2	3.083	1.182	5.353	3.489	1.896	2.017	1.967	1.864
1.18	6	1	5.067	2.269	[illegible]171	4.335	4.625	4.290	4.423	4.084
1.19(a)	5	12	3.084	12.243	65.317	33.704	17.608	12.635	9.519	8.683
1.19(b)	5	3	2.550	11.855	15.200	8.693	5.511	3.598	3.307	3.535
1.20	4	1	2.900	1.636	1.140	1.236	1.232	1.117	1.180	1.355
1.21	4	1	1.867	0.821	1.141	1.078	1.075	1.002	1.078	1.368
1.22	5	3	3.100	11.922	11.718	6.747	3.869	3.180	3.861	3.841
1.23	2	7	1.600	0.858	0.733	0.886	0.815	0.412	0.588	0.561
1.24	5	2	4.017	1.404	4.347	2.922	1.553	1.621	1.740	1.639
1.25	3	1	3.417	1.042	3.079	2.319	2.242	2.316	2.264	2.256
1.26(a)	7	0	5.833	14.849	0.000	0.567	0.044	0.546	0.363	0.272
1.26(b)	6	2	4.700	14.767	6.332	3.967	2.203	2.134	2.086	2.049
1.27	5	7	3.334	0.972	23.683	13.153	7.084	5.436	4.296	3.757
1.28	5	4	4.266	1.787	22.706	12.155	6.204	5.686	3.428	3.779
1.29	5	10	2.250	11.961	30.085	16.035	8.484	6.314	5.212	4.070
1.30	4	6	3.566	3.627	26.434	14.155	7.666	4.907	4.994	3.621
1.31	3	141	2.033	11.402	289.563	146.111	73.482	50.220	37.720	31.326
1.32	3	2	2.200	1.805	4.353	2.964	1.835	1.848	1.769	1.880
1.33	6	1	5.567	12.145	18.086	9.407	9.544	9.702	9.524	10.763
1.34	5	2	2.783	0.971	10.455	5.874	3.256	3.454	3.410	3.716
1.35	7	1	3.650	0.774	2.031	1.720	1.684	1.971	1.776	1.998
1.36	7	1	3.267	1.224	1.789	1.704	1.463	1.655	1.637	1.677
1.37	3	1	3.533	2.167	4.714	3.179	3.038	3.042	2.766	3.360
1.38	3	1	2.000	0.832	1.405	1.258	1.114	1.122	1.193	1.344
1.39	4	1	3.800	1.627	1.381	1.316	1.360	1.265	1.287	1.300
1.40	5	1	3.034	2.577	1.406	1.196	1.076	1.105	1.063	1.345
1.41	8	4	5.584	13.081	25.877	14.338	7.414	6.166	4.195	4.347
1.42	2	1	2.900	1.192	1.429	1.264	1.210	1.237	1.202	1.281
1.43	4	4	3.767	1.826	18.997	10.426	5.314	4.845	3.205	3.220
1.44	3	3	2.166	0.911	3.821	2.381	2.217	1.142	1.042	1.180
1.45	4	1	1.950	0.998	1.048	1.114	1.001	1.128	1.163	1.086
1.46	2	1	1.850	1.085	2.457	1.751	1.673	1.782	1.725	1.974
Mean	4.532	6.915	3.431	4.696	20.231	10.855	6.166	4.892	4.182	3.930
Median	5	2	3.084	1.826	4.353	2.964	2.217	2.254	2.111	2.049
StdDev	1.797	20.920	1.493	5.411	45.862	23.042	11.537	7.928	6.098	5.145

H. SUMMARY

MARIE has been designed and built using the client-server model to keep the size of each process small, to enable parallel processing, and to enhance modularity and reusability. Enhancing the system to

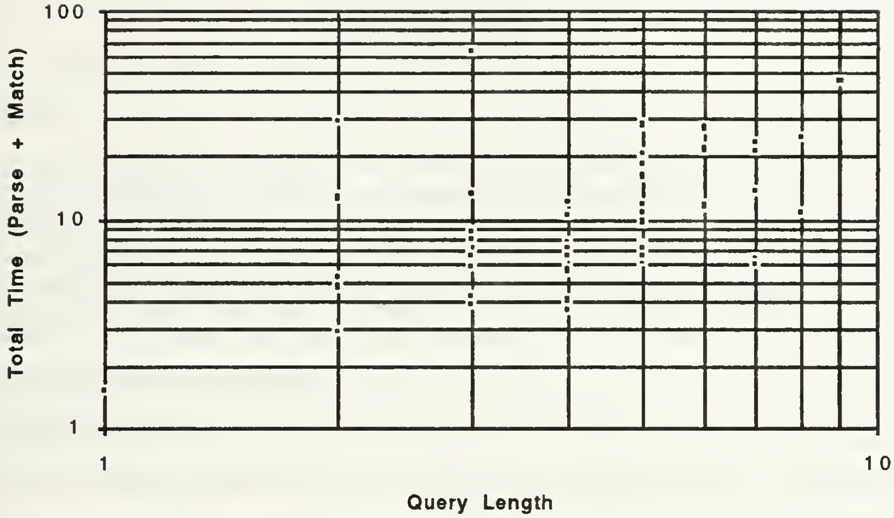

Figure 6.7 Log-Log Chart for Photo Lab queries (Query Length vs. Total Time).

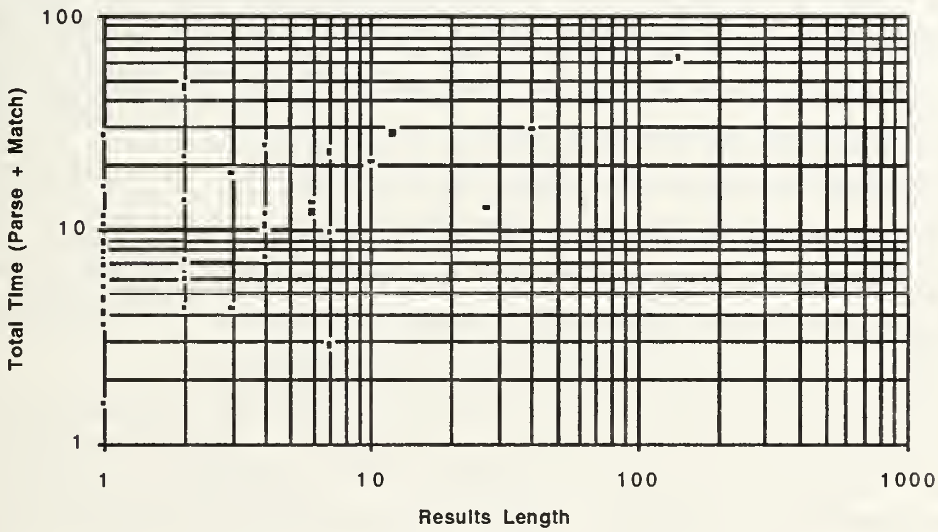

Figure 6.8 Log-Log Chart for Photo Lab queries (Results Length vs. Total Time).

recognize different potential patterns during the matching is a matter of adding the appropriate rules. There are many open questions on measuring the retrieval effectiveness, such as, determining how far we can go with a neutral logical form before being forced to use predefined scripts and frames to capture a certain event or what guidance can we provide to the user to improve retrieval. These issues confront all IR researchers in trying to find ways to improve retrieval effectiveness.

VII. CONCLUSION

A. INTRODUCTION

The ability to remotely access photographic images from a centralized database is now becoming a reality because of the Naval Air Warfare Center Weapons Division, China Lake's extensive networking capabilities and computing platforms. We have demonstrated an intelligent IR system for retrieving multimedia data using NL queries and SQL commands for potential use at the Center. Limitations of the keyphrase retrieval system pose the major hindrance to providing effective retrieval capabilities. The ability to use NL for query specification holds the most promise and provides the greatest challenges. We have shown that the system offers increased retrieval effectiveness over the keyphrase approach together with the ability to examine the multimedia data.

B. STRENGTHS AND WEAKNESSES

In the course of this research, we have observed that captions at the Center are very poorly written, grammatically speaking, and that they contain very rich, descriptive noun phrases. As a result of this work, we can provide suggestions on how better to write captions for retrievals, suggestions that can be incorporated into routine caption editing. Captions also use very few determiners, unlike dialog conversation, which relies more heavily on them. This fact is brought out in the original grammar rules where there are more determiner rules than any other category with the possible exception of verb and verb phrase rules. We know what changes we have made to handle written queries and captions, plus we also have the original rules to handle spoken dialog. We should be able to combine these two methodologies in the future to arrive at a system that can handle both written and spoken queries.

There are some immediate areas of concern confronting us. We have discovered that a solely syntactic parsing approach will not work for captions. The effectiveness of using grammar-rule weights in many cases is questionable and the combination of weights and case information for guiding the parsing is still guesswork. Deriving a semantic representation using a neutral structure would be ideal if we could guarantee a correct neutral structure all the time. Thus far, we have found this not to be the case. Hence, although the parsing works for the present test cases, we cannot guarantee that they will work for all captions.

The ability to parallelize the matching operation through the use of multiple fine-grain match processes on existing workstation resources gives us some means to control response time. Future investment in parallel computers may provide the opportunity to parallelize the coarse-grain match. One possible approach would be to use a shared memory parallel computer. Suppose that the *CgsQueue* was implemented in a shared memory and that for each keyword in the query, we also created separate hash

tables. Then we could assign processors to generate the union of all of the caption identifiers associated with a keyword class, its subclasses, and related fields from the relational database; the union is computed by the "OR" operation. Once a keyword's hash table was updated by all of a keyword's processors, we could then use the table to increment the *CgsQueue* via a simple increment. However, both the *CgsQueue* and individual keyword hash tables may not have the same caption identifier in the same position due to the order in which collisions occurred. Thus, we may need a more elaborate mechanism to increment the *CgsQueue* or a different means to record a caption identifier reference. We may also be able to use such shared memory and/or message-passing systems to accomplish the fine-grain match.

C. POSSIBLE EXTENSIONS

1. Unknown Words

One of the areas for future research is the handling of words and concepts that are not listed in the lexicon and type hierarchy. The need for automatic creation of the lexicon and type hierarchy is crucial, and was an issue brought out by Braun and Schwind (1976). Haas (1991) raised some concerns with respect to sublanguage technical terms and the fact that some of these terms might not be found in a general or machine-readable dictionary. Distinguishing between too many word senses was another concern. One approach to solving this problem follows. By loading the caption logical forms back into the type hierarchy and accumulating a count on the number of times that we have similar patterns (using the *case_vals* slot), we can have the system cache template graphs of class interactions. When undefined words are encountered, we can have the syntactic parser return the most promising parse tree for the surrounding phrase by substituting in different parts of speech (we make the assumption that undefined words are most likely to be nouns or verbs at the start). We can then proceed to create a possible logical form structure (graph) and then attempt to match this new graph against the existing predefined templates. In this case, we are trying to find the correspondence between the undefined word and one that may exist in the templates. This process can be viewed as analogical matching. The basic idea for applying this concept to defining unknown words was described by Berwick (1983). Jacobs and Zernik (1988) have proposed a gradual method to defining new words for the lexicon by analyzing a sequence of example text, making hypotheses, and refining the hypotheses.

2. NL Processing

As we pointed out in Chapter IV and the preceding section, there are many situations that the current NLP subsystem does not handle well; examples include quantifiers, conjunctions, preposition chains, incorrect parses for words having both noun and verb forms, etc. We believe that some additional semantic information is needed to derive a better syntactic representation. If our eventual goal is to construct

case frame representations for the logical form, then it may prove advantageous to move these case representations into the syntactic parse phase as a further guide to parsing.

The lexicon, once incorporated with our type hierarchy, can be developed into a useful representation language for Prolog applications, in the direction of KL-ONE [Brachman 89] or KRYPTON [Brachman 87b]. Also, deciding how much domain type information such as frame-recognition and frame-application rules to incorporate into this knowledge base to handle the repertoire of queries we can expect is still an open issue. We do not use frames and/or scripts in the current system because the queries we have encountered did not require them, but we did use the *infers* slot with its associated rules that provides some basic frame and script processing. Further use of MARIE by the retrieval analyst and other Center users may require such knowledge structures be incorporated to produce better matches as queries become more general instead of referencing specific terms (e.g., "missile" versus "AIM-9R") and more focused on obvious relationships (e.g., "missile is under a wing on an aircraft"). This type of evolution, however, will result in increased maintenance unless we have some means to make the system more intelligent.

3. User Models

The ability to capture retrieval characteristics about a user has been stressed in a number of systems. In Section 2, we described the PLEXUS expert system. An additional feature of this system is a primitive user model that captures a user's familiarity with the subject area in a way similar to what a reference librarian would try to do by asking the user various questions about him-/her-self. Such information can then be used by various modules as needed, the HELP facility being one example. Another approach to a user model is found in IR-NLI II [Brajnik 87]. A user model and expert system were developed to serve as a front-end to commercially available information retrieval systems. The user-model maintains both long- and short-term information (e.g., preferences and responses from previous query sessions and the current session) about the user when formulating a search strategy. Useful information to be captured in the user model is identified, however, the extent to which this information can be used in searching and storing information in the database is limited. Specifying search parameters is restricted to the interface provided by the commercial retrieval system and many such systems provide no means to modify the stored information using the model information.

In our application, a user model could be used to record the different types of events or concepts a user is often interested with. Events are described by verbs, and even though the captions we encountered used predominately nouns, there were many verbal type nouns used. Hence, we could see a greater emphasis placed on verbs by either translating the verbal nouns to events or inferring events from prepositional relationships. This increased use of verbs might then provide justification to include verbs in the coarse-grain match.

By also recording the queries and the captions selected from search results, we might be able to set or adjust both the coarse- and a fine-grain match thresholds for future queries, instead of relying on thresholds based solely on the number of logical form records.

4. Genetic Algorithms

Forrest [85] demonstrated the feasibility of representing complex knowledge structures using the classifier system, although her research contribution was not in information retrieval. She demonstrated how a knowledge representation scheme, KL-ONE, based on semantic networks could be translated into a classifier system. We anticipate that we can represent the stored caption and query structures in an analogous method. We believe we can use a genetic algorithm in performing the match in an approach similar to Gordon [88]. In addition, instead of modifying keywords, we can modify the caption knowledge structure representation (for single-user systems) and/or user model to reflect more accurately the way a user queries the captions.

D. SUMMARY

It is still too soon to evaluate our decision to use captions or the effectiveness of the processing strategy we have chosen. However, we feel that we now have a tangible system that can be demonstrated and built upon not only for images but also for other forms of multimedia data. Increased usage together with lessons learned from this research will allow us to fine-tune the system in the future.

REFERENCES

[Abramson 63] Abramson, N. 1963. *Information Theory and Coding.* New York, NY: McGraw-Hill Book Co., Inc.

[Allen 87] Allen, J. 1987. *Natural Language Understanding.* Menlo Park, CA: Benjamin Cummings Publishing Co.

[Alterman 85] Alterman, R. A Dictionary Based on Concept Coherence. 1985. *Artificial Intelligence* 25. pp. 153-186.

[Andre 88] Andre, E.; Herzog, G.; Tist, T. 1988. On the Simultaneous Interpretation of Real World Image Sequences and their Natural Language Description: The System SOCCER. *Proc., 8th European Conf. on AI* (ECAI 88). Munich, W.Germany. pp. 449-454.

[Anick 91] Anick, P. G. 1991. Integrating "Natural Language" and Boolean Query: An Application of Computational Linguistics to Full-text Information Retrieval. *AAAI-91 Natural Lanuage Text Retrieval Workshop.* Anaheim, CA. July 15.

[Baudin 91] Baudin, C., Gevins, J., Mabogunje, A., and Baya, V. 1991. A Knowledge-based Interface for Design Information Retrieval. *AAAI-91 Intelligent Multimedia Interfaces Workshop.* Anaheim, CA. July 15.

[Bertino 88] Bertino, E., Rabbiti, F., and Gibbs, S. 1988. Query Processing in a Multimedia Document System. *ACM Transactions in Office Information Systems* 6 (no. 1). January. pp. 1-41.

[Berwick 83] Berwick, R. C. Learning Word Meanings From Examples. 1983. *Proc., 8th Intl. Joint Conf. on AI* (IJCAI-83). Karlsruhe, Germany. pp 459-461.

[Boy 91] Boy, G. A. and Paris, C. L. 1991. An intelligent document browsing system that incorporates indexing in context. *AAAI-91 Intelligent Multimedia Interfaces Workshop.* Anaheim, CA. July 15.

[Brachman 83a] Brachman, R.J. 1983. What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *COMPUTER* 16 (no. 10). October. pp. 30-36.

[Brachman 83b] Brachman, R.J., Fikes, R.E., and Levesque, H.J. 1983. KRYPTON: A Functional Approach to Knowledge Representation. *COMPUTER* 16 (no. 10). October. pp. 67-73.

[Brachman 89] Brachman, R. J. and Schmolze, J. 1989. An Overview of the KL-ONE Knowledge Representation System. In *Readings In Artificial Intelligence and Databases.* Eds. J. Mylopoulos and M. Brodie. pp. 207-229. San Mateo, CA: Morgan Kaufmann Publ.

[Brajnik 87] Brajnik, G., Guida, G., and Tasso, C. 1987. User Modeling in Intelligent Information Retrieval. *Information Processing & Management* 23 (no. 4). pp. 305-320.

[Braun 76] Braun, S. and Schwind, C. 1976. Automatic, Semantics-Based Indexing of Natural Language Texts for Information Retrieval Systems. *Information Processing & Management* 12 (no. 1). pp. 147-153.

[Brooks 87] Brooks, H.M. 1987. Expert Systems and Intelligent Information Retrieval. *Information Processing & Management* 23 (no. 4). pp. 367-382.

[Briggs 85] Briggs, R. 1985. Knowledge Representation in Sanskrit and Artificial Intelligence. *AI Magazine* 6 (no. 1). Spring. pp 32-39.

[Chang 88] Chang, S., Yan, C., Dimitroff, D., and Arndt, T. 1988. An Intelligent Image Database System. *IEEE Transactions on Software Engineering* 14 (no. 5). May. pp 681-688.

[Charniak 87] Charniak, E.J. and McDermott, D. 1987. *Introduction to Artificial Intelligence*. Reading, MA: Addison-Wesley Publishing Co.

[Christodulakis 88] Christodulakis, S. and Graham, S. 1988. Browsing within Time-Driven Multimedia Documents. In *Office Information Systems* (Conference sponsored by ACM SIGOIS and IEEE CS TC-OA. Palo Alto, CA. March 23-25). Ed. R. Allen. pp. 219-227. New York, NY: ACM.

[Clifton 90] Clfiton, C. and Garcia-Molina, H. 1990. Indexing in a Hypertext Database. In *Proceedings of the Sixteenth Int'l Conf on Very Large Databases*. Brisbane, Australia. Aug 13-16. Eds. D. McLeod, R. Sacks-Davis, and H. Schek. pp. 36-49. Palo Alto, CA: Morgan Kaufman.

[Cohen 87] Cohen, P.R. and Kjeldsen, R. 1987. Information Retrieval by Constrained Spreading Activation in Semantic Networks. *Information Processing & Management* 23 (no. 4). pp. 255-268.

[Croft 87] Croft, W. B. Approaches to Intelligent Information Retrieval. 1987. *Information Processing & Management* 23 (no. 4). pp 249-254.

[Croft 88] Croft, W. B. and Krovetz, R. 1988. Interactive Retrieval of Office Documents. *Conference on Office Information Systems*. Palo Alto, CA. March 23-25. pp. 228-235.

[DTIC 90] *Defense Technical Information Center Thesaurus*. Defense Logistics Agency. Cameron Station. Alexandria, VA. AD-A226000. Sept. 1990.

[Damier 88] Damier, C. and Defude, B. 1988. The Document Management Component of a Multimedia Data Model. *Research and Development in Information Retrieval, 11th Intl. Conf.*. Grenoble, France. Ed. Y. Chiaramella. pp. 451-464. New York, NY: ACM.

[Davis 91] Davis, M. E. 1991. Director's Workshop: Semantic Video Logging with Intelligent Icons. *AAAI-91 Intelligent Multimedia Interfaces Workshop*. Anaheim, CA. July 15.

[Dick 91] Dick, J. P. and Hirst, G. 1991, Intelligent Text Retrieval. *AAAI-91 Natural Lanuage Text Retrieval Workshop*. Anaheim, CA. July 15.

[DiManzo 86] DiManzo, M., Adorni, G., and Giunchiglia, F. 1986. Reasoning about Scene Descriptions. *Proceedings of the IEEE* 74 (no. 7). July. pp. 1013-1025.

[Driscoll 91] Driscoll, J., D. Rajala, W. Shaffer, and D. Thomas. 1991. The Operation and Performance of an Artificially Intelligent Keywording System. *Information Processing & Management* 27 (no. 1). pp 43-54.

[Dulle 90] Dulle, J. 1990. *The Scope of Descriptive Captions for Use in a Multimedia Database System*. Master's Thesis. Naval Postgraduate School, Department of Computer Science, Monterey, CA.

[Dumais 88] Dumais, S.T., Furnas, G.W., and Landauer, T.K. 1988. Using Latent Semantic Analysis to Improve Access to Textual Information. *Proc., Conf on Human Factors in Computing Systems* (CHI'88). Washington, DC. pp. 281-285.

[Epstein 88] Epstein, S.S. 1988. Principle-based Interpretation of Natural Language Quantifiers. *7th Natl. Conf. on AI* (AAAI-88). Saint Paul, MN. pp. 718-723.

[Findler 79] Findler, N. 1979. A Heuristic Information Retrieval System Based on Associative Networks. In *Associative Networks: The Representation and Use of Knowledge by Computers*. Ed. N.V. Findler. pp. 306-326. New York, NY: Academic Press.

[Finin 86] Finin, T. W. 1986. Constraining the Interpretation of Nominal Compounds in a Limited Context. In *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*. pp 163-173. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

[Finin 91] Finin, T. W., McEntire, R., Weir, C., and Silk, B. 1991. A Three-Tiered Approach to Natural Language Text Retrieval. *AAAI-91 Natural Lanuage Text Retrieval Workshop*. Anaheim, CA. July 15.

[Frixione 89] Frixione, M., Gaglio, S., and Spinelli, G. 1989. Are there individual concepts? Proper names and individual concepts in SI-nets. *Intl. Journal Man-Machine Studies* 30. pp. 489-503.

[Gallant 91] Gallant, S. I. 1991. Context Vector Representations for Document Retrieval. *AAAI-91 Natural Lanuage Text Retrieval Workshop*. Anaheim, CA. July 15.

[Garey 79] Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W.H.Freeman and Co.

[Gordon 88] Gordon, M. 1988. Probabilistic and Genetic Algorithms for Document Retrieval. *Communications of the ACM* 31 (no. 10). October. pp. 1208-1218.

[Grosz 87] Grosz, B.J., Appelt, D.E., Martin, P.A., and Pereira, C.N. 1987. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. *Artificial Intelligence* 32. pp. 173-243.

[Guglielmo 92] Guglielmo, E. J., Nkwocha, N., and Rowe, N.C. 1992. A Prototype Decision Support System and Multimedia Database for Command and Control. In *9th Annual Conference on Command and Control Decision Aids*. Monterey, CA. June 8-10.

[Haas 90] Haas, S. W. 1991. A Feasibility Study of the Case Hierarchy Model for the Construction and Porting of Natural Language Interfaces. *Information Processing & Management* 26 (no. 5). pp. 615-628.

[Haas 91] Haas, S. W. 1991. Unknown Words: Is the Choice of Dictionary Important in Text Retrieval. *AAAI-91 Natural Lanuage Text Retrieval Workshop*. Anaheim, CA. July 15.

[Hahn 86] Hahn, U. and Reimer, U. 1986. TOPIC Essentials. In: *COLING 86: Proc 10th Int. Conf. on Computational Linguistics*. Bonn, Beligum. Aug 25-29. pp. 497-503.

[Hahn 88] Hahn, U. and Reimer, U. 1988. Automatic Generation of Hypertext Knowledge Bases. *Conference on Office Information Systems*. Palo Alto, CA. March 23-25. pp. 182-188.

[Hendrix 78] Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D. and Slocum, J. 1978. Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems* 3 (no. 2). June. pp. 105-147.

[Hilster 91] de Hilster, D. and Meyers, A. 1991. Heuristic Skimming of Voluminous Text. *AAAI-91 Natural Language Text Retrieval Workshop*. Anaheim, CA. July 15.

[Holtkamp 90] Holtkamp, B. and Lum, V. Y. 1990. *Integration of Alphanumeric and Media Data*. Internes Memorandum des Lehrstuhls Software-Technologie, Universitat Dortmund, Germany. Memo Nr. 48. July.

[Jacobs 88] Jacobs, P. and Zernik, U. 1988. Acquiring Lexical Knowledge from Text: A Case Study. *7th Natl. Conf. on AI* (AAAI-88). Saint Paul, MN. pp. 739-744.

[Jacobs 90] Jacobs, P.S. and Rau, L.F. 1990. SCISOR: A System for Extracting Information from On-Line News. Schenectady, NY: GE Research and Development Center. March 21.

[Kaplan 84] Kaplan, S.J. 1984. Designing a Portable Natural Language Database Query System. *ACM Transactions on Database Systems* 9 (no. 1). March. pp. 1-19.

[Katz 88] Katz, B. 1988. *Using English for Indexing and Retrieving*. AI Memo 1096. MIT, Cambridge, MA.

[Kolodner 83] Kolodner, J.L. 1983. Indexing and Retrieval Strategies for Natural Language Fact Retrieval. *ACM Transactions on Database Systems* 8 (no. 3). September. pp. 434-464.

[Lum 89] Lum, V.Y. and Meyer-Wegener, K. 1989. *A Multimedia Database Management System Supporting Contents Search in Media Data*. Rpt. No. NPS52-89-020. Naval Postgraduate School, Department of Computer Science, Monterey, CA.

[Lum 90] Lum, V. Y. and K. Meyer-Wegener. 1990. An Architecture for a Multimedia Database Management System Supporting Content Search. In *Advances in Computing and Information, Proceedings of the International Conference on Computing and Information*. Niagara Falls, Canada. May 23-26.

[Mauldin 91] Mauldin, M. L. 1991. *Conceptual Information Retrieval: A Case Study in Adaptive Partial Parsing*. Norwell, MA: Kluwer Academic Publishers.

[Mohan 88] Mohan, L. and Kashyap, R. 1988. An Object-Oriented Knowledge Representation for Spatial Information. *IEEE Transactions on Software Engineering* 14 (no. 5). May. pp. 675-681.

[Montgomery 89] Montgomery, C. A., J. Burge, H. Holmback, J. L. Kuhns, B. G. Stalls, R. Stumberger, and R. L. Russel Jr. 1989. The DBG Message Understanding System. In *Proceedings of the Annual AI Systems in Government Conference*. Washington, DC. March 27-31.

[NASA 88] *NASA Thesaurus, Hierarchical Listing*. NASA Scientific and Technical Information Division. NASA SP-7064. 1988.

[Nagel 88] Nagel, H. 1988. From Image Sequences towards Conceptual Descriptions. *Image and Vision Computing* 6 (no. 2). May. pp. 59-74.

[Neumann 83] Neumann, B. and Novak, H-J. 1983. Event Models for Recognition and Natural Language Description of Events in Real-World Image Sequences. *Proc., 8th Intl. Joint Conf. on AI* (IJCAI-83). Karlsruhe, Germany. pp. 724-726.

[Neumann 84] Neumann, B. 1984. Natural Language Access to Image Sequences: Event Recognition and Verbalization. *Proc., 1st Conf. on AI Appl.* Denver, CO. pp. 226-231.

[Niemann 84] Niemann, H., Bunke, H., Hofmann, I., and Sagerer, G., 1984. Diagnostic Inferences from Image Sequences - A Knowledge Based Approach. *Proc., 1st Conf. on AI Appl.* Denver, CO. pp. 610-615.

[Novak 86] Novak, H-J. 1986. Generating a Coherent Text Describing a Traffic Scene. *Proc., 11th Intl. Conf. on Computational Linguistics* (COLING-86). pp. 570-575.

[NSF 91] NSF9119 - *Science and Technology Information System User's Guide*. OIS. Oct 4, 1991.

[Ragusa 90] Ragusa, J. and G. Orwig. 1990. Attacking the Information Access Problem with Expert Systems. *Journal of Expert Systems* 4. Winter. pp. 26-32.

[Rau 87] Rau, L.F. 1987. Knowledge Organization and Access in a Conceptual Information System. *Information Processing & Management* 23 (no. 4). pp. 269-284.

[Roussopoulos 88] Roussopoulos, N. Faloutsos, C., and Sellis, T. 1988. An Efficient Pictorial Database System for PSQL. *IEEE Transactions on Software Engineering* 14 (no. 5). May. pp. 639-650.

[Rowe 88] Rowe, N.C. 1988. *Artificial Intelligence through Prolog*. Englewood Cliffs, NJ: Prentice-Hall.

[Rowe 91] Rowe, N. C. and Guglielmo, E. J. 1991. *Exploiting Captions for Access to Multimedia Databases*. Report No. NPSCS-91-012. Naval Postgraduate School, Department of Computer Science, Monterey, CA. April.

[Sager 86] Sager, N. 1986. Sublanguage: Linguistic Phenomenon, Computational Tool. In *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc. pp 163-173.

[Schank 75] Schank, R. 1975. *Conceptual Information Processing*. New York, NY: Elsevier Science Publishing Co.

[Schank 77] Schank, R. 1977. *Scripts, Plans, and Goals*. Hillsdale, NJ: Lawrence Erlbaum Associates.

[Schank 79] Schank, R. and Carbonell, J.G. 1979. RE: The Gettysburg Address, Representing Social and Political Acts. In In *Associative Networks: The Representation and Use of Knowledge by Computers*. Ed. N.V. Findler. pp. 327-362. New York, NY: Academic Press.

[Schirra 87] Schirra, J., Bosch, G., Sung, C., and Zimmerman, G. 1987. From Image Sequences to Natural Language: A First Step toward Automatic Perception and Description of Motion. *Applied Artificial Intelligence* 1 (no. 4). pp. 287-305.

[Sembok 90] Sembok, T. M. T. and van Rijsebergen, C. J. 1990. SILOL: A Simple Logical-Linguistic Document Retrieval System. *Information Processing & Management* 26 (no. 1). pp. 111-134.

[Smith 89] Smith, P.J., Shute, S.J., Galdes, D., and Chignell, M.H. 1989. Knowledge-Based Search Tactics for an Intelligent Intermediary System. *ACM Transactions on Information Systems* 7 (no. 3). July. pp. 246-270.

[Sowa 84] Sowa, J.F. 1984. *Conceptual Structures.* Reading, MA: Addison-Wesley Publishing Co.

[Stanfill 86] Stanfill, C. and Kahle, B. 1986. Parallel Free-Text Search on the Connection Machine System. *Communications of the ACM* 29 (no. 12). December. pp. 1229-1239.

[Velardi 88] Velardi, P., Pazienza, M.T., and De'Giovanetti, M. 1988. Conceptual Graphs for the Analysis and of Sentences. *IBM J. Res. Develop.* 32 (no. 2). March. pp. 251-267.

[Vickery 87] Vickery, A. and Brooks, H.M. 1987. PLEXUS - The Expert System for Referral. *Information Processing & Management* 23 (no. 2). pp. 99-117.

[Wilensky 86] Wilensky, R. 1986. Knowledge Representation - A Critique and a Proposal. In *Experience, Memory, and Reasoning.* Eds. J.L.Kolodner and C.K.Reisbeck. pp. 15-28. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

[Wong 87] Wong. S.K.M, Ziarko, W., Raghavan, V.V., and Wong, P.C.N. 1987. On Modeling of Information Retrieval Concepts in Vector Spaces. *ACM Transactions on Database Systems* 12 (no. 2). June. pp. 299-321.

[Zellweger 88]. Zellweger, P. 1988. Active Paths through Multimedia Documents. *Document Manipulation and Typography, Proc. of the Intl. Conf. on Electronic Publishing.* Nice, France. Ed. J. VanVliet. pp. 19-34. New York, NY: Cambridge University Press.

APPENDIX A

ORIGINAL CAPTIONS

124. facilities, harvey field, view of runway with aircraft from inside tower. personnel working.

3204. e1424, three f3d-1 aircraft on flightline. two of the aircraft carry sidewinder 1 missiles.

5628-30. air to air, sparrow missile firing from f6f aircraft. missile on aircraft, just igniting with plume and exhaust showing, and missile away from aircraft and smoke covering bottom of aircraft. wing of aircraft in view. excellent.

5824. air to air, f3h-1 bu# 133550 aircraft with china lake on tail. full side view with sidewinder 1 missile aboard.

10851. bullpup missile on bomb skid in hangar bay just outside of elevator aboard the uss lexington cva-16.

10862. sparrow iii, guardian of the skies, operational with seventh fleet in western pacific. four missiles on underside of f3h-1 bu# 137010, point mugu 7010 on tail. air to air view from side during firing of one missile.

10880. graphics art. artist conception of fbm guided missile submarine ssg-n1, nuclear program. submarine firing missile.

29263. tim, tracking instrumentation mount, in machine shop at michelson lab. fully assembled with camera and operator.

29293. air to air, ya-4c bu# 145063 aircraft, china lake on tail, firing hipeg mk 4 gun pod. excellent side view.

29773. sidewinder 1a and 1c missiles. comparison of 1a aim 9b, 1c irah aim 9d and 1c sar aim 9c, 3/4 right side view on stand.

34070. air to air, a-4b bu# 142777 aircraft, china lake, with snipe missile. rounds 3 and 4 on centerline station with multiple bomb rack configuration. viewed from underside of aircraft.

34271. fireye {clam b} bombs after drop from an a-4e aircraft. excellent picture of target in the midst of the receeding fireball.

38239. photographic equipment. extended range tracking mount on trailer.

40226. adam search set pod on s-2a aircraft. pod at 17 degrees tilt down.

41136. htw, helicopter trap weapon, multiple drop sequence from uh-1e helicopter. parachutes opened on four weapons.

43176. air to air, walleye fat albert on a-4c bu# 147781 aircraft, china lake on tail. full side view of

aircraft with dummy missile over b-1-b target center.

43694. rockeye ii bombs on an a-6a aircraft, nose 17, no other markings, with wings in folded position. bombs mounted on wings and centerline station.

44263-67. shrike firing at scr 584 shrike bullpup target. series, split frame 35mm data film, shows 3/4 view of missile approaching and destroying target.

45935-37. local wildflowers. series shows beavertail cactus in bloom, dalea in bloom and closeup of beavertail cactus flower.

62439-41. pictorial, kern river views. closeup of large rocks showing water erosion, looking down river with small girl on rock, and river in foreground with trees and mountain range in background.

64287-89. air to air, phoenix missile impacting qf-9 drone. large explosion with drone still in view. f-14a tomcat launching missile, missile has dropped and motor has fired and still underneath aircraft. missile in flight with motor burning and plume behind.

64472-74. heliostat, direct solar intensity studies. target hoisting with crane, preparing target for hoisting, target on top of tower.

65019. wild horse in the snow on coso range. one horse with blaze, facing camera.

66694-96. airfield naf, hangar #1 with view of aircraft flight line, cold line, and hot line, overview. looking nw from top of hangar #3.

69812. photovoltaic cell panels for generating power to ultimately operate a radar. l to r, nasa employee and richard fulmer with the batteries and power inverter.

85486-89. project 163. rapec seat ejection over t-5 g-1 range from qf-9f drone aircraft. views showing seat firing, dummy clearing aircraft, end of rocket burn, and parachute deployment of pilot and seat. ship target #e on ground under aircraft.

85486. project 163: rapec seat ejection i. (seat firing), with ship target #e in background. f9f.

85487. project 163: rapec i seat ejection 2. (clearing the aircraft), with ship target #e. f-9f.

85488. rapec i seat ejection 3. (end of burning.)

85489. rapec i seat ejection 4. (parachute deployment of pilot and seat.)

91572. tim, tracking instrumentation mount, in machine shop at michelson lab. fully assembled with camera and operator.

110169. bat and standard arm missiles with a-6a bu# 151562, nose #562, ng on tail. excellent side view closeup of missiles, standard arm mounted on wing with bat missile sitting on ground beneath.

161044-45. bqm drone mounted on dc-130a bu# 158228 vc-3 aircraft, closeup view of drone and aircraft wing. bqm control helicopter uh-2a bu# 149033 usn. full side view.

161044. bqm control helicopter: side view of uh-2a bu# 149033 usn.

161045. bqm mounted on a c130 aircraft: close side view of drone. dc-130a bu# 158228.

161082-84. air to air, sidewinder laim {aim 91}, launch from hh-1k helicopter, coming in on tank target. viewed from two directions.

161082. laim (sidewinder aim 91) launch from hh-1k helicopter air to air firing

163030. fae weapons on av-8a bu# 158389 harrier aircraft, vma-513 usmc, nose 6, wf on tail. 3/4 front overall view with hangar 1 in background.

164800-05. shrike missile on a-7e bu# 157525 aircraft, nose #306, aj on tail. closeup and overall views.

164803. shrike: missile on a-7e aircraft

164804. shrike: missile on a-7e aircraft

166318. agile-quickturn missile ftv-2 leaving launcher. excellent view of missile with plume, clear of launcher.

168578-79. walleye ii program aboard the uss kitty hawk in the gulf of tonkin. closeup of walleye ii missile and data link pod on a-7e aircraft. flight deck crew in view.

168578. walleye ii: data link walleye ii on a-7e bu# 1575

174921. condor missile, dev-assist #1, video view as missile warhead explodes. sequence composite. excellent full side view.

178010-13. roseville incident. sco-96 mk 81 bomb electric heat cookoff test 454692. bomb wrapped and wires attached. pre test view.

178012. roseville test #454692. mk 81 bomb, sco 96, cookoff. view of bomb wrapped and wires attached before test.

180657. e3923, av-8a harrier, 600 keas, escape system test, rd 4. synchro firing at 4505'n x 34'e, camera 44. dummy just leaving cockpit with seat rockets burning. chute in air unfilled. debris in air.

181709-10. flir mod a-7, pod installed on a-7e bu# 156734 aircraft in hangar 3.

181709. flir mod a7, flir pod installation a7e aircraft in hangar # 3.

181754. expendable seeker simulator, ess, installed in pod on a-6a aircraft. full side view of pod with personnel.

181761-70. roseville incident. sco-146 dodx boxcar cookoff with 24 inert bombs on pallet inside with clock. interior damage and exterior view of boxcar at bldg 63f.

181761. boxcar test, sco 146, roseville incident. cookoff of inert bombs at 63f.

182711-13. air to air, sidewinder aim 9l, ext-024, firing on qf-86 drone over g range. missile several 100 feet out, touching nose and drone exploding.

182711. air to air view of sidewinder aim 9l firing on qf-86 drone over g range. ext-024.

182712. air to air view of sidewinder aim 9l firing on qf-86 drone over g range. ext-024.

182713. air to air view of sidewinder aim 9l firing on qf-86 drone over g range. ext-024.

183531. harpoon missile launch aboard uss high point pch-1, hydrofoil underway in puget sound, and firing one harpoon. excellent port side view during firing.

185854-68. rotor wing dispenser spin canister on uh-1n helicopter bu# 158771. closeup views, views of r. w. strongback, fmc spin canister, and overall view of helicopter with canisters.

185854. rotary wing dispenser spin canister on uh-1n helicopter

185860. rotary wing dispenser spin canister on uh-1n helicopter

185864. rotary wing dispenser r. w. strongback on uh-1n a/c

185866. rotary wing dispenser: fmc spin canister on uh-1n a/c

188716. program echo. load test on xn-01 pedestal structure, 45 deg, elevation, front view.

209362. e9937, assault breaker martin marietta warhead sled test, synchro firing at 166'n x 35'w. sub-missiles released.

209862. air to air, vx-5 aircraft on sinkex operation. a-6e, a-7e, a-7e, and a-4m carrying lgb's, laser guided bombs. excellent view over desert terrain with bank of clouds in background.

210192-99. phoenix tdd dsu-28/b standard missile ex-62, tdd flyover test. nra-3b bu# 142667, nose 71, pmtc, in flight. views from ground at different altitudes.

210192. phoenix fuze flyover test, view of a3 aircraft, #71, flying over test setup.

210453-63. air to air, a-4m bu# 160264 skyhawk ii aircraft, 2nd maw/marines, with two laser maverick agm-65c usaf training missiles flying over high sierra's. excellent shots, side and 3/4 rear views.

210455. a4m #160264, skyhawk ii, aircraft, air to air, with two laser maverick agm-65c training missiles flying over high sierra's. excellent shots, side view.

210593. sea launch of trident missile 1979, eastern test range.

213528-29. nwc range control center construction progress, front and side view of building with antenna tower complete.

213528. nwc range control center construction progress, front view of building with antenna tower complete.

213529. nwc range control center construction progress, side view of building with antenna tower complete.

213795-99. hytas flight test on uh-1n, misty 13, helicopter. inside view of instrumentation, close up front view of helicopter. helicopter lifting off with excellent view of hangar and control tower in background.

213795. hytas flight test on uh1n, misty 13, helicopter. inside view of helicopter.

213798. hytas flight test on uh1n, misty 13, helicopter. helicopter lifting off with excellent view of hangar and control tower in background.

213799. hytas flight test on uh1n, misty 13, helicopter. close up front view of helicopter.

213853-57. lgb, skipper, bomb on mk 7 loader and loading on a-7c aircraft bu# 156739, cl on tail. views of aircraft on runway.

213853. bomb, lgb "skipper" on mk 7, dolly loader.

213855. bomb, lgb "skipper" being loaded from mk 7 loader to a7c aircraft. personnel loading.

213856. bomb, lgb "skipper" on a7c aircraft. bu # 156739, 1/4 front view on runway.

213857. bomb, lgb "skipper" on a7c aircraft. bu # 156739, side view on runway.

215669. tp1314, a-7b/e dvt-7, siiis-er, 250 keas, escape system, run 2, synchro firing at 1090'n x 38'w, dummy just leaving sled.

216381-82. f-15 aircraft firing a sidewinder aim 9m missile against qf-86 target. view of qf-86 and impact.

216382. f-15 aircraft firing a sidewinder 9m missile against qf-86 target. view of qf-86 and impact.

216383. air to air, f-14a bu# 157990 tomcat, nose 211, pmtc on tail, with phoenix missile xaim-54c, edm-4.

217936-41. air to air, f-4b aircraft bu# 149451 with china lake on tail, over sierra nevada mountains.

217938. f-4b aircraft, bu # 9451 with china lake on tail. air to air view over sierra mtns.

217942-49. e1346, martin marietta, csws dispenser, assault breaker, rd 1, synchro firing at 3119'n x 19'e, 3189'n x 19'w, 3268'n x 19'e, 3295'n x 24'e, 3417'n x 32'w, and 3456'n x 32'w, various views of sled on track with plume showing and projectiles in air.

217947. e 1346, martin marietta assault breaker, rd1, synchro firing at 3377'n x 24'e, view of sled on track with plume showing and projectiles in air.

218178-89. uss forrestal, cva-59. fire accident, action views. uss repertus dd-851 and helicopter standing by to assist crewmen fighting fire, examining burned remains. capt john d. beling inspecting damage to flight deck. photo's taken in gulf of tonkin on 07/29/67.

218178. uss forrestal, cva-59. view showing smoke pouring from ship, uss repertus dd-851 and helicopter standing by to assist.

218179. uss forrestal, cva-59. view showing crewmen fighting burning aircraft aboard flight deck.

218183. uss forrestal, cva-59. view showing crewmen towing the smouldering ra-5c aircraft to the edge of the flight deck in an attempt to save the fire from spreading.

218184. uss forrestal, cva-59. view showing crewmen fighting fire aboard the flight deck.

218185. uss forrestal, cva-59. view showing crewmen examining gutted aircraft on flight deck.

218186. uss forrestal, cva-59. view showing stunned crewmen gazing at the dead bodies placed in hangar bay # 3.

218187. uss forrestal, cva-59. view showing crewmen examining the burned remains aboard flight deck.

218188. uss forrestal, cva-59. view showing capt john k beling, usn inspecting damage to flight deck.

218189. uss forrestal, cva-59. view showing crewmen relocatng badly burned aircraft.

218915. tp1356, a-7b/e escape system, siiis-er, srt-6, synchro firing at 107'n x 30'w. dummy just breaking canopy.

218997-99. lwir, long wavelength infrared, side view without target, strobe, type s. side view with mig-21 model target, mercury vapor lights, type s and l.

218997. lwir "long wavelength infared", side view without target. stroge, type s.

218999. lwir "long wavelength infared", side view with target. mercury vapor lights, type l.

219074-79. sidearm missile on av-8a bu# 158706 harrier aircraft, usmc vx-5, nose 27, xe on tail. various full views of aircraft with missile on stbd wing launcher.

219075. sidearm missile on av8 harrier aircraft, bu # 158706, marines. 1/4 front overall view. av-8a bu# 158706 aircraft, vx-5 usmc, nose 27, tail 27 xe 158706.

219079. sidearm missile on av8 harrier aircraft, bu # 158706, marines. overall rear view. av-8a bu# 158706 aircraft, vx-5 usmc, nose 27, tail 27 xe 158706.

219539-42. a/c survivability j52 fuel ingestion tp-13-80, dump test setup. sequences 1 - 4.

219539. a/c survivability j52 fuel ingestion tp-13-80, dump test setup. sequences 1 - 4.

219551-58. michelson lab soldering assembly area. personnel viewed soldering wires into connector, placing integrated circuit chip, soldering resistors, and inspection of self-solderability.

219551. soldering assembly area. soldering wires into connector.

219553. soldering assembly area. linda wincn placing integrated circuit chip on solder board.

219554. soldering assembly area. soldering resistor on g-r simulator circuit board.

219555. soldering assembly area. preparing to solder resistor on g-r simulator circuit board.

219557. soldering assembly area. soldering resistor on g-r simulator circuit board. richard maxwell.

219558. soldering assembly area. self-solderability inspection in soldering assembly area. linda wincn.

219907-08. air to ari, air force f-4g "wild weasel" usaf# 69263 with weapons load top to bottom, harm agm-88, maverick agm-65d ir usaf, alq-119 ecm pod, standard arm, and shrike agm-45. side view with sierra mountains in background.

219907. air to air of air force f-4g "wild weasel" tail # 263 with weapons load, top to bottom, harm agm-88, maverick agm-65, alq-119 ecm pod, standard arm, and shrike agm-45. side view with sierra nevada mountains in background.

220152. honeywell gun, 120mm projectile at 3740' per second, in front of fireball. excellent shot.

220748. air to air, a-7e BU# 160857 aircraft, china lake on tail, dropping mk 82 bombs over nwc coso bombing range covered with snow. pilot lcdr r. kapernick.

221353-54. tp2162, sidearm missile, {modified sidewinder 9c} firing test from air force f4-g aircraft

against ground tank target with radar. view of missile as it approaches and reaches target.

221353. tp 2162, sidearm missile, "modified sidewinder 9c" firing test from air force f4g aircraft against ground tank target with radar. view of missile as it approaches target.

221354. tp 2162, sidearm missile, "modified sidewinder 9c" firing test from air force f4g aircraft against ground tank target with radar. view of missile as it reaches target.

223156. harm missile coming into radar van target at g-1 range. view just before impact.

224159-66. maverick agm-65c laser usaf coming in on tank target at eglin afb, impact and explosion. sequence views of firing, and before and after.

224159. laser mavericks hitting tank targets.

224161. laser mavericks hitting tank targets. view of target exploding.

224163. laser mavericks hitting tank targets. view of target after explosion.

224164. laser mavericks hitting tank targets. view of target and truck.

224547-54. bigeye sectioned weapon, blu-80/b binary chemical, tail fin, aft port, tail latch assembly, fwd port, potted, aft end closeups showing tail, mdf and booster of aft port.

224547. bigeye sectioned weapon, blu-80/b binary chemical, tail fin closed.

224548. bigeye sectioned weapon, blu-80/b binary chemical, aft port, sectioned.

224549. bigeye sectioned weapon, blu-80/b binary chemical, tail latch assembled, closed.

224550. bigeye sectioned weapon, blu-80/b binary chemical, fwd port, potted.

224551. bigeye sectioned weapon, blu-80/b binary chemical, total view of aft end, tail open.

224552. bigeye sectioned weapon, blu-80/b binary chemical, straight on view of aft bulkhead, tail open.

225194-97. air to air, av-8b bu# 161398 aircraft, v/stol, no nose #, 3 161398 marines on tail, production model with camouflaged paint and mk 82 inert practice bombs. views over natc pax river and missouri river.

225194. air to air, av-8b, v/stol usmc bu #161398, production model, with camouflaged paint, mk 82 inert practice bombs. full side view over pax river. av-8b bu# 161398 aircraft, no nose #, 3 161398 marines on tail, no other markings.

225196. air to air, av-8b, v/stol usmc bu #161398, production model, with camouflaged paint, mk 82 inert practice bombs. 1/4 overhead view over missouri river and farm land. av-8b bu# 161398 aircraft, no nose #, 3 161398 marines on tail, no other markings.

226385-86. program f/a-18a. major jon gallinetti at the controls of the f/a-18 simulator at wssa, hangar 3. excellent. second view with personnel standing behind.

226385. program f/a-18a. major jon gallinetti at the controls of the f/a-18a simulator. excellent.

226386. program f/a-18a. major jon gallinetti at the controls of the f-18 simulator. other personnel in

view also.

227282-86. samson decoy drones uploaded on iter #11 launcher on f/a-18a aircraft by personnel. drones being unloaded from aero 51a trailer. two front views of drones on aircraft. first view has f-4j bu# 158378 aircraft, sh on tail, in background.

227282. samson drones being loaded onto an f-18 aircraft, pre flight. view with personnel unloading drones from aero 51a trailer. f-4j bu# 158378, sh on tail in background.

227283. samson drones being loaded onto an f-18 aircraft, pre flight. ordnance crew loading samson drone on aircraft.

227284. samson drones being loaded onto an f-18 aircraft, pre flight. loaded on iter #11 launcher on f-18.

227285. samson drones being loaded onto an f-18 aircraft, pre flight. view of two samson drones on aircraft.

227286. samson drones being loaded onto an f-18 aircraft, pre flight. front view of two drones on aircraft.

227462. air to air, f/a-18a bu# 161366 aircraft, nose 1, tail 1, pilot major gallinetti, usmc, over olancha peak with snow still on ground.

228544-46. acimd, advanced common intercept missile demonstration, missile on stand in front of mich lab. full side and 1/4 front and side view with kennedy plaque on rock at rear.

228544. acimd "advanced common intercept missile demonstration" missile on stand in front of mich lab. full side view.

228545. acimd "advanced common intercept missile demonstration" missile on stand in front of mich lab. 1/4 front view.

228546. acimd "advanced common intercept missile demonstration" missile on stand in front of mich lab. side view.

228795-97. phoenix missile intercept of qf-48 #01, second flight of drone. views showing missile coming into aircraft, impact with shaped charge explosion, and aircraft after hit.

228795. phoenix missile intercept of qf-4b no. 01, second flight of drone. view showing aircraft after hit. large plume from aircraft.

228796. phoenix missile intercept of qf-4b no. 01, second flight of drone. view showing aircraft just prior to impact of missile.

228797. phoenix missile intercept of qf-4b no. 01, second flight of drone. view showing aircraft impact, shaped charge and plume.

230834. f/a-18a bu# 161720 aircraft, side 102, flying eagle on tail, with harm missile, over sea site, closeup of aircraft with center of sea site in view.

231373-75. air to air, f/a-18a bu# 161720, nose 102, over nwc ranges. harm, pod & b-61 or mk 91 silver bullet uploaded. just before, during and with harm missile in front of aircraft with diamond plume.

231373. air to air of harm missile launched from f-18 aircraft, bu # 161720, over nwc ranges. excellent view just before firing.

231374. air to air of harm missile launched from f-18 aircraft, bu # 161720, over nwc ranges. excellent full side view after firing with missile just in front of aircraft. excellent diamond pattern in missile exhaust.

231375. air to air of harm missile launched from f-18 aircraft, bu # 161720, over nwc ranges. excellent side view at fring. excellent diamond pattern in missile exhaust.

232744-47. air to air, f/a-18a bu# 161720 aircraft, nose 102, with harm missile and b57-vfa-82a camera pod. various views of plane over nwc ranges, some during banking. one full frame view.

232744. air to air view of f/a-18 aircraft with harm missile and b57-vfa-82a pod. side view of plane over nwc range mountains.

232745. air to air view of f/a-18 aircraft with harm missile and b57-vfa-82a camera pod. side view of plane during banking over nwc ranges. note wash in center of shot.

232747. air to air view of f/a-18 aircraft with harm missile and b57-vfa-82a camera pod. side view of plane during banking over nwc ranges. full frame of plane.

234050. weapons survivability j52 fuel ingestion fireball. excellent.

234056-71. lllgb, low level laser guided bomb, gbu-22/b on stand. various identification views, fins open and closed. excellent.

234064. lllgb "low level laser guided bomb", gbu-22/b on stand. left side view, fins open.

234756. air to air, qf4h-1 drone, bu# 149452. drone painted silver with red on nose, under wings, and on tail and fin. full side view. almost complete view of armitage field above aircraft. excellent.

237492. tp85t221, tav-8b dvt-2r, 225 kts seat ejection at 1900'n x 21'w. rear dummy ejecting, drogue chute above but not opened. seat rockets firing.

238225. na-3b bu# 142630 skywarrior aircraft with standard missile ii{n}, ex-62 tdd in nose for cft. front view of plane with ex-62 installed. radome not installed.

238226. air to air, na-3b aircraft bu# 142630 with standard missile ii{n}, ex-62 tdd in nose for cft. full frame view of aircraft over north range of nwc. excellent.

239091-98. wssa, f/a-18a validation lab. validation cockpit station overall view, center console, john hessler on left and capt. s. germain, canadian, on right, avionics racks and computer peripherals.

239091. wssa, f/a-18 validatoin lab. validation cockpit station, overall view.

239092. wssa, f/a-18 validation lab. validaton cockpit station, center console.

239093. wssa, f/a-18 validatoin lab. validaton cockpit station with john hessler on left and capt. s.

germain, canada on right.

239094. wssa, f/a-18 validaton lab. validatoin cockpit station with capt. stephen germain, canada.

239095. wssa, f/a-18 validatoin lab. validation cockpit station with john hessler.

239097. wssa, f/a-18 validatoin lab. validation cockpit station avionics racks.

239098. wssa, f/a-18 validaton lab. validaton cockpit station computer peripherals.

239126. air to ground, program na-3b. aar-47 electro optical device testing onna-3b aircraft bu# 142630 over armitage. na-3b center, all three hangars in view, looking west. excellent view of field and mountains.

241425-29. sidearm missile approaching target, pole with checkerboard, tank and radar. page 63, aviation week of 01/20/86. three views of sidearm impacting target.

241426. sidearm missile approaching pole and checker board target, tank and radar in front of target.

241427. sidearm missile impacting target, pole with checkerboard, tank and radar.

241451-52. av-8b harrier ii aircraft, nose 622, marine on tail, full side view.

241452. av-8b "harrier" aircraft, side view. av-8b aircraft, nose 622, marine on tail. no other markings.

241950. sidearm missile firing from ah-1j helicopter. missile away from aircraft with plume and aircraft in view.

242099. sidewinder aim 9l opnav evaluation. missile launched from vx-5 supercobra ah-1w, missile in front of helo with large plume behind. over g-2 range.

242109. air to air, harm missile agm-88{i} launch from ea-6b aircraft over nwc range. full side view before launch.

242112. air to air, harm missile agm-88{i} launch from ea-6b aircraft over nwc range. missile at ignition with plume alongside aircraft, missile just leaving launcher.

247152-55. air to air, two a-7e's armed with mk 82 bombs, flying over coso range and over sierra's.

247152. air to air of two a-7e's armed with mk 82 bombs, flying over coso range with sierra's in background.

247153. air to air of two a-7e's armed with mk 82 bombs, flying over coso range.

247155. air to air of two a-7e's armed with mk 82 bombs, flying over sierra's.

247181-88. sidewinder seeker 9r with and without dome.

247181. sidewinder seeker 9r without dome.

247186. sidewinder seeker 9r with dome.

247740-41. f/a-18a bu# 162396 aircraft, nose 105, china lake, with shrike missile over g range. full side and close bottom view from side. full side view.

247740. aircraft fa-18, bu# 162396, with shrike missile over g range. full side view.

247741. aircraft fa-18, bu# 162396, with shrike missiles over g range. full side view.

248387-92. bomb mk 82 h-6 test 1 fast cookoff. pre and post views.

248387. bomb mk 82 h6 test #1 fast cookoff. pre test.

249254. cruise tomahawk missile exploding over f-4 target in revetment. f-4 on fire with annular blast overhead. excellent.

250629-30. air to air, na-3b bu# 142630 aircraft outfitted with tacit/rainbow knozy, closeup of fwd part of aircraft and overall view of entire aircraft. tp88a031.

250629. air to air, na-3b bu# 142630 aircraft outfitted with tacit/rainbow knozy, closeup of fwd part of aircraft with nose clearly shown. tp88a031.

250630. air to air, na-3b bu# 142630 aircraft outfitted with tacit/rainbow knozy, overall view of entire aircraft. tp88a031.

251272. tp88t052, naces t-45a goshawk seat ejection run 1. synchro firing at 4597'n x 51'w. fire plume from pusher sled, both seat rockets with fire plume and dummy over sled with drogue chute extended. excellent.

251700-10. tp86a057, air to air, f/a-18c bu# 163428 aircraft, nose 110, flying eagle on tail, with a agm-65e laser maverick and two sidewinder aim 9's inert, various views over argus and coso mountains with storm cloud cover,

251701. tp 86a057, air to air, f/a-18c bu# 163428 aircraft with a agm-65e laser maverick and two sidewinder aim 9's [inert] making left bank away from camera. excellent. f/a-18c bu# 163428 aircraft. nose 110. 10 and flying eagle on tail. s under cockpit.

251703. tp 86a057, air to air, f/a-18c bu# 163428 aircraft with a agm-65e laser maverick and two sidewinder aim 9's [inert] making right bank towards camera. excellent cloud formatoins in background. f/a-18c bu# 163428 aircraft. nose 110, 10 and flying eagle on tail. s under cockpit.

251704. tp 86a057, air to air, f/a-18c bu# 163428 aircraft with a agm-65e laser maverick and two sidewinder aim 9's [inert] over argus range. panamint valley in view with cloud cover above. excellent. f/a-18c bu# 163428 aircraft. nose 110. 10 and flying eagle on tail. s under cockpit.

251706. tp 86a057, air to air, f/a-18c bu# 163428 aircraft with a agm-65e laser maverick and two sidewinder aim 9's [inert] over argus range. pretty cloud cover. f/a-18c bu# 163428 aircraft. nose 110. 10 and flying eagle on tail. s under cockpit.

251707. tp 86a057, air to air, f/a-18c bu# 163428 aircraft with a agm-65e laser maverick and two sidewinder aim 9's [inert] in level flight. f/a-18c bu# 163428 aircraft. nose 110. 10 and flying eagle on tail. s under cockpit.

251708. tp 86a057, air to air, f/a-18c bu# 163428 aircraft with a agm-65e laser maverick and two sidewinder aim 9's [inert] over coso range. side view. f/a-18c bu# 163428 aircraft. nose 110. 10 and flying

eagle on tail. s under cockpit.

251709. tp 86a057, air to air, f/a-18c bu# 163428 aircraft with a agm-65e laser maverick and two sidewinder aim 9's [inert] over coso range. vertical frame. f/a-18c bu# 163428 aircraft. nose 110. 10 and flying eagle on tail. s under cockpit.

251710. tp 86a057, air to air, f/a-18c bu# 163428 aircraft with a agm-65e laser maverick and two sidewinder aim 9's [inert] over coso range. full frame view. f/a-18c bu# 163428 aircraft. nose 110. 10 and flying eagle on tail. s under cockpit.

251855. helicopter th-1l bu# 157834, nose 014, taking off, left side view with personnel in view on helo.

252492. tp87a209, sidewinder aim 9m firing from f/a-18a aircraft at drone bqm-34s. f/a-18a carrying two sidewinder aim 9m's. underneath full side view.

252494. tp87a209, sidewinder aim 9m firing from f/a-18a aircraft at drone bqm-34s. f/a-18a has just fired missile, missile with large plume and in front of aircraft. excellent.

252496-97. tp87a209, sidewinder aim 9m firing from f/a-18a aircraft at drone bqm-34s. f/a-18a has fired missile, missile in front of aircraft with smoke trail extending back behind aircraft. excellent.

252496. tp87a209, sidewinder aim 9m firing from f/a-18a aircraft at drone bqm-34s. f/a-18a has fired missile, missile in front of aircraft with smoke trail extending back behind aircraft. excellent.

253959-63. air to air, aircraft ta-7c bu# 156738, 700 on nose. project skyray. various views of aircraft with hs camera station 6, walleye btv {ballistic test vehicle} station 7, and skyray fiber optics instrumentation pod station 8. mk 82 bomb {inert} on left wing.

253959. air to air, aircraft ta-7c bu #156738, 700 on nose. fiber optics instrumentation pod and ballistic test vehicle "btv" under wing, closeup right side view.

253960. air to air, aircraft ta-7c bu #156738, 700 on nose. viewed from below, station 6 hs camera, station 7 walleye "btv" ballistic test vehicle, station 8 skyray instrumentation pod.

253961. air to air, aircraft ta-7c bu #156738, 700 on nose. full side view of aircraft with station 6 hs camera, station 7 "btv" ballistic test vehicle, station 8 instrumentation pod. note haze over valley in background. walleye btv skyray instrumentation pod.

253962. air to air, aircraft ta-7c bu #156738, 700 on nose. full side view of aircraft with station 6 hs camera, station 7 "btv" ballistic test vehicle, station 8 instrumentation pod, and mk 82 bomb (inert) on left wing. walleye btv skyray instrumentation pod.

255577-80. tp89a053, air to air, sidewinder aim 9m separation test. f/a-18a bu# 161362 aircraft, nose 101, with sidewinder aim 9m on right wing tip. fuel dumping from tail fins. views over searles valley, g ranges, and ridgecrest. excellent.

255578. tp89a053, air to air. aim-9m separation test. f/a-18a, bu# 161713 nose 101, with sidewinder

aim-9m on right wing tip. fuel dumping from tail. searles lake in background.

255652-55. asroc vla rocket motor fsq-12 sn 031 post static firing disassembly at bldg 69.

255655. asroc vla rocket motor fsq-12 sn 031 post static firing disassembly at bldg 69.

256393-95. sidewinder aim 9r et-2 and sidewinder aim 9l mounted on f/a-18a bu# 162396 aircraft, nose 105, tail 05 with flying eagle. aim 9r on outboard pylon and aim 9l on inboard pylon. closeup full front, 1/4 front and full view of missiles and rear of aircraft.

256393. sidewinder aim 9r et-2 and sidewinder aim 9l mounted on an f/a-18 aircraft. aim 9r on outboard pylon and aim 9l on inboard pylon. closeup front view. f/a-18a bu# 162396 aircraft, nose 105. tail 05 with flying eagle.

256395. sidewinder aim 9r et-2 and sidewinder aim 9l mounted on an f/a-18 aircraft. aim 9r on outboard pylon and aim 9l on inboard pylon. full view of missiles and rear of aircraft. f/a-18a bu# 162396 aircraft, nose 105. tail 05 with flying eagle.

256979-99. program skyray, f/a-18a bu# 162396 aircraft, nose 105, china lake, with rtv skyray pod. various views of overall of aircraft and closeup of pod. beautiful cloud formations behind aircraft. shot at gun butts.

256999. program skyray. f/a-18a bu# 162396 aircraft, nose 105, china lake, with rtv skyray pod, closeup 3/4 rear view of pod.

257009-19. av-8b night attack bu# 162966 aircraft, nose 87 night hawk logo, tail 162966 marines, mad `marine air detachment'. sunset and special lighting. side and closeup of nose and cockpit, side view, excellent.

257019. av-8b night attack bu# 162966 aircraft, nose 87, marine air detachment. sunset and special lighting. closeup of nose and cockpit, side view, excellent. av-8b bu# 162966 aircraft, nose 87 with night hawk logo & night attack, 162966 and marines on tail.

257055. graphics composite. f/a-18a aircraft firing sidewinder aim 9r et-1. aircraft piloted by cdr benjes. intercept with qf-86 drone on april 12, 1989.

257110-19. a-6e bu# 161084 aircraft, vx-5, xe on tail, nose 22, night attack, preparing for takeoff with sunset over the sierras. pilot lt j. a. rzeszotko and bn seat lt j. c. weston. excellent 3/4 front view with cloud formations.

257110. a-6e bu# 161084 aircraft, vx-5, xe on tail, nose 22, night attack, preparing for takeoff with sunset over the sierras. pilot lt rzeszotko and back seat lt weston. excellent 3/4 front view.

257135. f/a-18a night attack aircraft, vx-5, nose 34, tail 34 xe, pilot cdr j. c. antonio with night vision goggles on. close port side view of pilot in cockpit. excellent.

257274-76. air to air, four plane formation, ea-6b vx-5, a-7e nwc, a-6e nwc, and f/a-18a nwc. all aircraft loaded with agm-88 harm. various views in clouds.

257274. air to air four plane formation, left to right clockwise, vx-5's ea-6b, nwc's a-7e, a-6e, and f/a-18a. all aircraft loaded with agm-88 harm. overhead right side view.

258795. night vision goggles. f/a-18a aircraft, vx-5, nose 35, uploaded with harm, maverick and sidewinder missiles. cdr j. c. antonio in cockpit with night vision goggles turned on. full side view just after sunset.

264966-68. f/a-18 firing sidewinder aim 9r, and intercepting qf-86 drone. copied from video screen.

262865-73.

262865. sidewinder aim 9r missile on stand, 3/4 front view.

262866. sidewinder aim 9r missile on stand, front view.

262867. sidewinder aim 9r misille on stand, 3/4 front view.

262868. sidewinder aim 9r missile mounted on f/a-18c bu# 163284 aircraft, nose 110. closeup side view of missile on outboard wing pylon.

262869. sidewinder aim 9r missile mounted on f/a-18c bu# 163284 aircraft, nose 110. closeup front view of missile on outboard wing pylon.

262870. sidewinder aim 9r missile mounted on f/a-18c bu# 163284 aircraft, nose 110. closeup view of front of missile and launcher.

262871. sidewinder aim 9r missile mounted on f/a-18c bu# 163284 aircraft, nose 110. 3/4 front view of aircraft.

262872. sidewinder aim 9r missile mounted on f/a-18c bu# 163284 aircraft, nose 110. 3/4 front view of aircraft.

262873. sidewinder aim 9r missile mounted on f/a-18c bu# 163284 aircraft, nose 110. full side view of aircraft.

264968. f/a-18 firing sidewinder aim 9r. missile just intercepting qf-86 drone. copied from video screen.

900304-05. air to air, northrop atb 'advanced technology {steaolth} bomber', b-2 aircraft, first public flight late 1989.

APPENDIX B

MODIFIED CAPTIONS

124. facilities at Harvey Field. view of runway with aircraft from inside tower. personnel working.

3204. TP 1424. three F3D-1 aircraft on flightline. two aircraft carry Sidewinder missiles.

10851. Bullpup missile on bomb skid in hangar bay just outside of elevator aboard the USS Lexington CVA-16.

10862. Sparrow III (Guardian of the Skies) operational with Seventh Fleet in western Pacific. four missiles on underside of F3H-1 BU# 137010 aircraft (Point Mugu 7010 on tail). air to air view from side during firing of one missile.

10880. graphics art. artist conception of FBM guided missile submarine SSG-N1 (nuclear program). submarine firing missile.

29263. TIM (Tracking Instrumentation Mount) in Machine Shop at Michelson Lab. TIM fully assembled with camera and operator.

29293. air to air view of YA-4C BU# 145063 aircraft (China Lake on tail) firing a Hipeg MK 4 gun. excellent side view.

29773. Sidewinder 1A and Sidewinder 1C missiles. comparison of Sidewinder 1A AIM 9B, Sidewinder 1C IRAH AIM 9D, and Sidewinder 1C SAR AIM 9C. 3/4 right side view on stand.

34070. air to air view of A-4B BU# 142777 aircraft (China Lake) with Snipe missile. round 3 on centerline station with multiple bomb rack. view of missile from underside of aircraft.

34271. Fireye (CLAM B) bombs after drop from an A-4E aircraft. excellent picture of target in the midst of the receeding fireball.

38239. photographic equipment. extended range tracking mount on trailer.

40226. Adam search set pod on S-2A aircraft. pod at 17 degrees tilt (down).

41136. HTW (Helicopter Trap Weapon) multiple drop sequence from UH-1E helicopter. parachutes opened on four weapons.

43176. air to air view of Walleye Fat Albert on A-4C BU# 147781 aircraft (China Lake on tail). full side view of aircraft with dummy bomb over B-1-B target center.

43694. Rockeye II bombs on an A-6A aircraft (nose 17 and no other markings) with wings in folded position. bombs mounted on wings and centerline station.

44263. Shrike firing at SCR 584 Shrike Bullpup target. 3/4 view of missile approaching and destroying target.

45935. local wildflowers. Beavertail cactus in bloom.

45936. local wildflowers. Dalea in bloom.

45937. local wildflowers. closeup view of Beavertail cactus flower.

5824. air to air view of F3H-1 BU# 133550 aircraft with China Lake on tail. full side view with Sidewinder missile aboard aircraft.

62439. pictorial. Kern River views. closeup view of large rocks showing erosion by water.

62440. pictorial. Kern River views. looking down river. small girl on rock.

62441. pictorial. Kern River views. river in foreground. trees and mountain range in background.

64287. air to air view of Phoenix missile impacting QF-9 drone. large explosion with drone still in view. F-14A Tomcat launching missile.

64288. air to air view of Phoenix missile impacting QF-9 drone. large explosion with drone still in view. F-14A Tomcat launching missile. missile has dropped and motor has fired. missile still underneath aircraft.

64289. air to air view of Phoenix missile impacting QF-9 drone. large explosion with drone still in view. F-14A Tomcat launching missile. missile in flight. motor burning. plume behind aircraft.

64472. heliostat. direct solar intensity studies. target hoisted with crane.

64473. heliostat. direct solar intensity studies. preparing target for hoisting.

64474. heliostat. direct solar intensity studies. target on top of tower.

65019. wild horse in the snow on Coso Range. one horse with blaze is facing camera.

66694. NAF airfield. hangar 1 with view of aircraft flightline.

66695. NAF airfield. overview of cold line and hot line.

66696. NAF airfield. looking NW from top of hangar 3.

69812. photovoltaic cell panels for generating power to ultimately operate a radar. left to right: NASA employee and Richard Fulmer with the batteries and power inverter.

85486. project 163. RAPEC seat ejection over T-5 G-1 Range from QF-9F drone aircraft. view showing seat firing. ship target #E in background.

85487. project 163. RAPEC seat ejection over T-5 G-1 Range from QF-9F drone aircraft. view showing dummy clearing aircraft. ship target #E on ground under aircraft.

85488. project 163. RAPEC seat ejection over T-5 G-1 range from QF-9F drone aircraft. view showing end of rocket burn.

85489. project 163. RAPEC seat ejection over T-5 G-1 range from QF-9F drone aircraft. view showing parachute deployment of pilot and seat.

110169. Bat and Standard Arm missiles with A-6A BU# 151562 aircraft (nose 562 and NG on tail). excellent closeup side view of missiles. Standard Arm mounted on wing. Bat missile sitting on ground beneath wing.

161044. USN UH-2A BU# 149033 BQM control helicopter (full side view).

161045. BQM drone mounted on DC-130A BU# 158228 VC-3 aircraft. closeup view of drone and aircraft wing.

161082. air to air view of Sidewinder LAIM (AIM 9L) launch from HH-1K helicopter. missile coming in on tank target.

163030. FAE weapons on USMC VMA-513 AV-8A BU# 158389 Harrier aircraft (nose 6 and WF on tail). 3/4 front overall view with hangar 1 in background.

164803. Shrike missile on A-7E BU# 157525 aircraft (nose 306 and AJ on tail). closeup view.

164804. Shrike missile on A-7E BU# 157525 aircraft (nose 306 and AJ on tail). overall view.

166318. Agile-Quickturn missile (FTV-2) leaving launcher. excellent view of missile with plume. missile clear of launcher.

168579. Walleye II program aboard the USS Kitty Hawk in the Gulf of Tonkin. closeup of Walleye II missile and data link pod on A-7E aircraft. flight deck crew in view.

174921. Condor missile (Dev-Assist 1) video view as missile warhead explodes. sequence composite.

178012. Roseville incident. SCO-96 MK 81 bomb electric heat cookoff test 454692. view of bomb wrapped. view of wires attached to bomb. pretest view.

180657. TP 3923. AV-8A Harrier (600 keas) escape system test (round 4). synchro firing at 4505'N x 34'E from camera 44. dummy just leaving cockpit. seat rockets burning. unfilled chute in air. debris in air.

181709. FLIR mod A-7. pod installed on A-7E BU# 156734 aircraft in hangar 3.

181754. Expendable Seeker Simulator (ESS) installed in pod A-6A aircraft. full side view of pod with personnel.

181761. Roseville incident. SCO-146 DODX boxcar cookoff test with 24 inert bombs on pallet inside of boxcar with clock. interior damage to boxcar. exterior view of boxcar at bldg 63.

182711. air to air view of Sidewinder AIM 9L firing at QF-86 drone over G Range. missile is several 100 feet out.

182712. air to air view of Sidewinder AIM 9L firing at QF-86 drone over G Range. missile touching nose.

182713. air to air view of Sidewinder AIM 9L firing at QF-86 drone over G Range. drone exploding.

183531. Harpoon missile launch aboard USS High Point PCH-1 hydrofoil underway in Puget Sound. hydrofoil firing one Harpoon. excellent hydrofoil port side view during firing.

185854. rotary wing dispenser spin canister on UH-1N BU# 158771 helicopter. closeup view.

185860. rotary wing dispenser spin canister on UH-1N BU# 158771 helicopter. overall view of helicopter with canisters.

185864. rotary wing dispenser spin canister on UH-1N BU# 158771 helicopter. view of R. W. Strongback on UH-1N aircraft.

185866. rotary wing dispenser spin canister on UH-1N BU# 158771 helicopter. view of FMC spin canister.

188716. program echo. load test on XN-01 pedestal structure (front view) with 45 degree elevation.

209362. TP 9937. Martin Marietta Assault Breaker warhead sled test. synchro firing at 166'N x 35'W. submissiles released.

209862. air to air view of VX-5 aircraft on Sinkex operation. A-6E, A-7E's, and A-4M carrying LGB's (Laser Guided Bombs). excellent view over desert terrain. bank of clouds in background.

210192. Phoenix TDD DSU-28/B Standard Missile EX-62. TDD flyover test. PMTC NRA-3B BU# 142667 aircraft (nose 71) in flight. view from ground.

210455. air to air view of A-4M BU# 160264 Skyhawk II aircraft (2nd MAW/Marines) with two laser Maverick AGM-65C USAF training missiles. aircraft flying over high Sierras. side view.

210593. sea launch (1979) of Trident missile at eastern test range.

213528. NWC Range Control Center construction progress. front view of building with antenna tower completed.

213529. NWC Range Control Center construction progress. side view of building with antenna tower completed.

213795. Hytas flight test on UH-1N (Misty 13) helicopter. inside view of helicopter instrumentation.

213798. Hytas flight test on UH-1N (Misty 13) helicopter. helicopter lifting off. excellent view of hangar and control tower in background.

213799. Hytas flight test on UH-1N (Misty 13) helicopter. closeup front view of helicopter.

213853. LGB Skipper bomb on MK 7 dolly loader.

213855. LGB Skipper bomb being loaded from MK 7 loader to A-7C BU# 156739 aircraft (CL on tail). personnel loading.

213856. LGB Skipper bomb on A-7C BU# 156739 aircraft (CL on tail). 1/4 front view of aircraft on runway.

213857. LGB Skipper bomb on A-7C BU# 156739 aircraft (CL on tail). side view of aircraft on runway.

215669. TP 1314. A-7B/E DVT-7 (250 keas) escape system test (run 2). synchro firing at 1090'N x 38'W. dummy just leaving sled.

216382. F-15 aircraft firing a Sidewinder 9M missile against QF-86 target. view of QF-86 and impact.

216383. air to air view of F-14A BU# 157990 Tomcat (nose 211 and PMTC on tail) with Phoenix missile XAIM-54C (EDM-4).

217938. air to air view of F-4B BU# 149451 aircraft (China Lake on tail) over Sierra Nevada

mountains.

217947. TP 1346. Martine Marietta Assault Breaker CSWS dispenser test (round 1). synchro firing at 3119'N x 19'E. view of sled on the track. plume showing and projectiles in air.

218178. USS Forrestal CVA-59 in Gulf of Tonkin. smoke pouring from ship. USS Repertus DD-851 and helicopter standing by to assist.

218179. USS Forrestal CVA-59 in Gulf of Tonkin. crewmen fighting burning aircraft aboard flight deck.

218183. USS Forrestal CVA-59 in Gulf of Tonkin. crewmen towing the smouldering RA-5C aircraft to the edge of the flight deck to stop the fire from spreading.

218184. USS Forrestal CVA-59 in Gulf of Tonkin. crewmen fighting fire aboard the flight deck.

218185. USS Forrestal CVA-59 in Gulf of Tonkin. crewmen examining the gutted aircraft on flight deck.

218186. USS Forrestal CVA-59 in Gulf of Tonkin. stunned crewmen gazing at the dead bodies placed in hangar bay 3.

218187. USS Forrestal CVA-59 in Gulf of Tonkin. crewmen examining the burned remains aboard flight deck.

218188. USS Forrestal CVA-59 in Gulf of Tonkin. Capt John Beling (USN) inspecting damage to flight deck.

218189. USS Forrestal CVA-59 in Gulf of Tonkin. crewmen relocatng badly burned aircraft.

218915. TP 1356. A-7B/E SRT-6 escape system test. synchro firing at 1075'N x 30'W. dummy just breaking canopy.

218997. LWIR (Long Wavelength Infrared) side view without target. strobe type s.

218999. LWIR (Long Wavelength Infrared) side view with MIG-21 model target. mercury vapor lights type l.

219075. Sidearm missile on USMC VX-5 AV-8A BU# 158706 Harrier aircraft (nose 27 and XE on tail) 1/4 front overall view of aircraft with missile on stbd wing launcher.

219079. Sidearm missile on USMC VX-5 AV-8A BU# 158706 Harrier aircraft (nose 27 and XE on tail). overall rear view of aircraft with missile on stbd wing launcher.

219539. a/c survivability. J52 fuel ingestion. TP 1380. dump test setup.

219551. the soldering assembly area in Michelson Lab. personnel soldering wires into connector.

219553. the soldering assembly area in Michelson Lab. Linda Wincn placing integrated circuit chip on solder board.

219554. the soldering assembly area in Michelson Lab. personnel soldering resistor on G-R simulator circuit board.

219555. the soldering assembly area in Michelson Lab. personnel preparing to solder the resistor on G-R simulator circuit board.

219557. the soldering assembly area in Michelson Lab. Richard Maxwell soldering resistor on G-R simulator circuit board.

219558. the soldering assembly area in Michelson Lab. self-solderability inspection in the soldering assembly area by Linda Wincn.

219907. air to air view of Air Foce USAF# 69263 F-4G Wild Weasel with weapons loaded top to bottom: Harm AGM-88, USAF IR Maverick AGM-65D, ALG-119 ECM pod, Standard Arm, and Shrike AGM-45. Sierra Nevada mountains in background.

220152. Honeywell gun. 120 mm projectile at 3740 feet per second in front of fireball.

220748. air to air view of A-7E BU# 160857 aircraft (China Lake on tail) dropping MK-82 bombs over NWC Coso Range covered with snow. pilot LCDR Kapernick.

221353. TP 2162. Sidearm missile (modified Sidewinder 9C) firing test from Air Force F4G aircraft against ground tank target with radar. view of missile as it approaches target.

221354. TP 2162. Sidearm missile (modified Sidewinder 9C) firing test from Air Force F4G aircraft against ground tank target with radar. view of missile as it reaches target.

223156. Harm missile coming into radar van target at G-1 Range. view of missile just before the impact.

224159. USAF AGM-65C laser Mavericks hitting tank targets at Eglin AFB.

224161. USAF AGM-65C laser Mavericks hitting tank targets at Eglin AFB. view of target exploding.

224163. USAF AGM-65C laser Mavericks hitting tank targets at Eglin AFB. view of target after explosion.

224164. USAF AGM-65C laser Mavericks hitting tank targets at Eglin AFB. view of target and truck.

224547. Bigeye sectioned weapon (BLU-80/B binary chemical bomb). tail fin closed.

224548. Bigeye sectioned weapon (BLU-80/B binary chemical bomb). aft port.

224549. Bigeye sectioned weapon (BLU-80/B binary chemical bomb). tail latch assembly closed

224550. Bigeye sectioned weapon (BLU-80/B binary chemical bomb). potted fwd port.

224551. Bigeye sectioned weapon (BLU-80/B binary chemical bomb). total view of bomb aft end. tail opened.

224552. Bigeye sectioned weapon (BLU-80/B binary chemical bomb). straight on view of bomb aft bulkhead. tail opened.

225194. air to air view of USMC AV-8B BU# 161398 V/STOL aircraft (Marines 161398 on tail).

production model with camouflaged paint and MK 82 inert practice bombs. full side view over Pax River.

225196. air to air view of USMC AV-8B BU# 161398 V/STOL aircraft (Marines 161398 on tail). production model with camouflaged paint and MK 82 inert practice bombs. 1/4 overhead view over Missouri River and farm land.

226385. program F/A-18A. Major Jon Gallinetti at the controls of the F/A-18A simulator at WSSA (hangar 3).

226386. program F/A-18A. Major Jon Gallinetti at the controls of the F/A-18A simulator at WSSA (hangar 3). view with personnel standing behind Gallinetti.

227282. Samson decoy drones being loaded onto an F/A-18A aircraft (preflight). view with personnel unloading drones from Aero 51A trailer. F-4J BU# 158378 aircraft with SH on tail in background.

227283. Samson decoy drones being loaded onto an F/A-18A aircraft (preflight). ordnance crew loading Samson drone on aircraft.

227284. Samson decoy drones being loaded onto an F/A-18A aircraft (preflight). drones loaded on ITER 11 launcher on F/A-18A.

227285. Samson decoy drones being loaded onto an F/A-18A aircraft (preflight). view of two Samson drones on aircraft.

227286. Samson decoy drones being loaded onto an F/A-18A aircraft (preflight). front view of two drones on aircraft.

227462. air to air view of F/A-18A BU# 161366 aircraft (nose 1 and tail 1). pilot Major Gallinetti (USMC) over Olancha Peak with snow still on ground.

228544. ACIMD (Advanced Common Intercept Missile Demonstration) missile on stand in front of Mich Lab. full side view with Kennedy plaque on rock at rear.

228545. ACIMD (Advanced Common Intercept Missile Demonstration) missile on stand in front of Mich Lab. 1/4 front view with Kennedy plaque on rock at rear.

228546. ACIMD (Advanced Common Intercept Missile Demonstration) missile on stand in front of Mich Lab. side view with Kennedy plaque on rock at rear.

228795. Phoenix missile intercept of QF-4B (number 1). second flight of drone. view of aircraft after hit. large plume from aircraft.

228796. Phoenix missile intercept of QF-4B (number 1). second flight of drone. view of aircraft just prior to impact of missile.

228797. Phoenix missile intercept of QF-4B (number 1). second flight of drone. view of aircraft impact with shaped charge explosion.

230834. F/A-18A BU# 161720 aircraft (side 102 and flying eagle on tail) with Harm missile over sea site. closeup view of aircraft and center of sea site.

231373. air to air view of Harm missile launched from F/A-18A BU# 161720 aircraft (nose 102) over NWC ranges. Harm, pod, and MK 91 Silver Bullet uploaded. excellent view of aircraft just before firing of Harm.

231374. air to air view of Harm missile launched from F/A-18A BU# 161720 aircraft (nose 102) over NWC ranges. Harm, pod, and MK 91 Silver Bullet uploaded. excellent full side view of aircraft after firing of missile with missile just in front of aircraft. excellent diamond pattern in missile exhaust.

231375. air to air view of Harm missile launched from F/A-18A BU# 161720 aircraft (nose 102) over NWC ranges. Harm, pod, and MK 91 Silver Bullet uploaded. excellent side view of aircraft at firing. excellent diamond pattern in missile exhaust.

232744. air to air view of F/A-18A BU# 161720 aircraft (nose 102) with Harm missile and B57-VFA-82A camera pod. side view of plane over NWC range mountains.

232745. air to air view of F/A-18A BU# 161720 aircraft (nose 102) with Harm missile and B57-VFA-82A camera pod. side view of plane during banking over NWC ranges. wash in center of view.

232747. air to air view of F/A-18A BU# 161720 aircraft (nose 102) with Harm missile and B57-VFA-82A camera pod. side view of plane during banking over NWC ranges. full frame of plane.

234050. weapons survivability. fireball from J52 fuel ingestion.

234064. LLLGB (Low Level Laser Guided Bomb) GBU-22/B on a stand. left side view. fins open.

234756. air to air view of QF4H-1 BU# 149452 drone. drone painted silver with red on nose, red under wings, red on tail, and red on fin. full side view. complete view of armitage field above aircraft.

237492. TP 85221. TAV-8B DVT-2R. seat ejection (225 kts) at 1900'N x 21'W. rear dummy ejecting. drogue chute is above dummy but is not opened. seat rockets are firing.

238225. NA-3B BU# 142630 Skywarrior aircraft with Standard Missile II-N and EX-62 TDD in nose for CFT. front view of plane with EX-62 installed. radome is not installed.

238226. air to air view of NA-3B BU# 142630 aircraft with Standard Missile II-N and EX-62 TDD in nose for CFT. full frame view of aircraft over north range of NWC.

239091. WSSA F/A-18A Validation Lab. validation cockpit station (overall view).

239092. WSSA F/A-18A Validation Lab. center console of validation cockpit station.

239093. WSSA F/A-18A Validation Lab. validation cockpit station with John Hessler on left and Capt Germain (Canada) on right.

239094. WSSA F/A-18A Validation Lab. validation cockpit station with Capt Stephen Germain (Canada).

239095. WSSA F/A-18A Validation Lab. validation cockpit station with John Hessler.

239097. WSSA F/A-18A Validation Lab. avionic racks of validation cockpit station.

239098. WSSA F/A-18A Validation Lab. computer peripherals of validation cockpit station.

239126. air to ground view of program NA-3B. AAR-47 electro optical device testing on NA-3B BU# 142630 aircraft over Armitage. NA-3B in center of view. all three hangars looking west. excellent view of field and mountains.

241426. Sidearm missile approaching target (pole with checkerboard). tank and radar are in front of target.

241427. Sidearm missile impacting target (pole with checkerboard). tank and radar.

241452. full side view of AV-8B Harrier II aircraft (nose 622 and Marines on tail).

241950. Sidearm missile firing from AH-1J helicopter. missile is away from aircraft. view of plume and aircraft.

242099. Sidewinder AIM 9L OPNAV evaluation. missile launched from VX-5 Super Cobra AH-1W over G-2 Range. missile in front of helo. large plume behind missile.

242109. air to air view of Harm AGM-88I missile launched from EA-6B aircraft over NWC range. full side view before launch.

242112. air to air view of Harm AGM-88I missile launched from EA-6B aircraft over NWC range. missile at ignition. plume alongside aircraft. missile just leaving launcher.

247152. air to air view of two A-7E"s armed with MK 82 bombs. aircraft flying over Coso Range. Sierras in background.

247153. air to air view of two A-7E"s armed with MK 82 bombs. aircraft flying over Coso Range.

247155. air to air view of two A-7E"s armed with MK 82 bombs. aircraft flying over Sierras.

247181. Sidewinder AIM 9R seeker without dome.

247186. Sidewinder AIM 9R seeker with dome.

247740. F/A-18A BU# 162396 aircraft (nose 105 and China Lake on tail) with Shrike missile over G Range. full side view of aircraft.

247741. F/A-18A BU# 162396 aircraft (nose 105 and China Lake on tail) with Shrike missile over G Range. close bottom view of aircraft.

248387. bomb MK 82 H-6 test 1, fast cookoff. pretest view.

249254. Tomahawk cruise missile exploding over F-4 target in revetment. F-4 on fire. annular blast is overhead.

250629. TP 88031. air to air view of NA-3B BU# 142630 aircraft outfitted with Tacit/Rainbow knozy. closeup view of fwd part of aircraft with nose clearly shown.

250630. TP 88031. air to air view of NA-3B BU# 142630 aircraft outfitted with Tacit/Rainbow knozy. overall view of entire aircraft.

251272. TP 88052. Naces T-45A Goshawk seat ejection (run 1). synchro firing at 4597" N x 51" W. fire plume from pusher sled. both seat rockets with fire plume. dummy over sled. drogue chute extended.

251701. TP 86057. air to air view of F/A-18C BU# 163428 aircraft (nose 110 and flying eagle on tail) with an AGM-65E laser Maverick and two Sidewinder AIM 9"s (inert). aircraft making left bank away from camera.

251703. TP 86057. air to air view of F/A-18C BU# 163428 aircraft (nose 110 and flying eagle on tail) with an AGM-65E laser Maverick and two Sidewinder AIM 9"s (inert). aircraft making right bank towards camera. excellent cloud formations in background.

251704. TP 86057. air to air view of F/A-18C BU# 163428 aircraft (nose 110 and flying eagle on tail) with an AGM-65E laser Maverick and two Sidewinder AIM 9"s (inert). aircraft over Argus Range. Panamint Valley in view with cloud cover above it.

251706. TP 86057. air to air view of F/A-18C BU# 163428 aircraft (nose 110 and flying eagle on tail) with an AGM-65E laser Maverick and two Sidewinder AIM 9"s (inert). aircraft over Argus Range. pretty cloud cover.

251707. TP 86057. air to air view of F/A-18C BU# 163428 aircraft (nose 110 and flying eagle on tail) with an AGM-65E laser Maverick and two Sidewinder AIM 9"s (inert). aircraft in level flight.

251708. TP 86057. air to air view of F/A-18C BU# 163428 aircraft (nose 110 and flying eagle on tail) with an AGM-65E laser Maverick and two Sidewinder AIM 9"s (inert). aircraft over Coso Range. side view.

251709. TP 86057. air to air view of F/A-18C BU# 163428 aircraft (nose 110 and flying eagle on tail) with an AGM-65E laser Maverick and two Sidewinder AIM 9"s (inert). aircraft over Coso Range. vertical frame view.

251710. TP 86057. air to air view of F/A-18C BU# 163428 aircraft (nose 110 and flying eagle on tail) with an AGM-65E laser Maverick and two Sidewinder AIM 9"s (inert). aircraft over Coso Range. full frame view.

251855. TH-1L BU# 157834 helicopter (nose 14) taking off. left side view with personnel on helo.

252492. TP 87209. Sidewinder AIM 9M firing from F/A-18A aircraft at drone BQM-34S. F/A-18A carrying two Sidewinder AIM 9M"s. full side view underneath aircraft.

252494. TP 87209. Sidewinder AIM 9M firing from F/A-18A aircraft at drone BQM-34S. F/A-18A has just fired missile. missile with large plume and in front of aircraft.

252496. TP 86209. Sidewinder AIM 9M firing from F/A-18A aircraft at drone BQM-34S. F/A-18A has fired missile. missile in front of aircraft. smoke trail extending back behind aircraft.

253959. program Skyray. air to air view of TA-7C BU# 156738 aircraft (nose 700). Skyray fiber optics instrumentation pod and Walleye BTV (Ballistic Test Vehicle) are under wing. closeup right side view of aircraft.

253960. program Skyray. air to air view of TA-7C BU# 156738 aircraft (nose 700). view below

aircraft. HS camera at station 6. Walleye BTV (Ballistic Test Vehicle) at station 7. Skyray instrumentation pod at station 8.

253961. program Skyray. air to air view of TA-7C BU# 156738 aircraft (nose 700). full side view of aircraft. HS camera at station 6. Walleye BTV (Ballistic Test Vehicle) at station 7. Skyray instrumentation pod at station 8. haze over valley in background.

253962. program Skyray. air to air view of TA-7C BU# 156738 aircraft (nose 700). full side view of aircraft. HS camera at station 6. Walleye BTV (Ballistic Test Vehicle) at station 7. Skyray instrumentation pod at station 8. MK 82 bomb (inert) on left wing.

255577. TP 89-A-053. air to air view of AIM-9M separation test. F/A-18A BU# 161713 aircraft, nose 101, with Sidewinder AIM-9M on right wingtip. fuel dumping from tail fins.

255578. TP 89053. air to air view of AIM-9M separation test. F/A-18A BU# 161713 aircraft (nose 101) with Sidewinder AIM-9M on right wingtip. ıuel dumping from tail fins. Searles Lake in background.

255580. TP 89-A-053. air to air view of AIM-9M separation test. F/A-18A BU# 161713 aircraft, nose 101, with Sidewinder AIM-9M on right wing tip. Ridgecrest in background.

255655. ASROC VLA FSQ-12 rocket motor (SN 031). motor post static firing disassembly at bldg 69.

256393. Sidewinder AIM 9R ET-2 and Sidewinder AIM 9L mounted on F/A-18A BU# 162396 aircraft, nose 105 and tail 5 with Flying Eagle. AIM 9R on outboard pylon and AIM 9L on inboard pylon. closeup front view.

256394. Sidewinder AIM 9R ET-2 and Sidewinder AIM 9L mounted on F/A-18A BU# 162396 aircraft, nose 105 and tail 5 with Flying Eagle. AIM 9R on outboard pylon and AIM 9L on inboard pylon. closeup 1/4 front view of front of missiles only.

256395. Sidewinder AIM 9R ET-2 and Sidewinder AIM 9L mounted on F/A-18A BU# 162396 aircraft, nose 105 and tail 5 with Flying Eagle. AIM 9R on outboard pylon and AIM 9L on inboard pylon. full view of missiles and rear of aircraft.

256979. program Skyray. F/A-18A BU# 162396 aircraft (nose 105 and China Lake on tail) with RTV Skyray pod. 3/4 stbd front view of pod.

256999. program Skyray. F/A-18A BU# 162396 aircraft (nose 105 and China Lake on tail) with RTV Skyray pod. closeup 3/4 rear view of pod. beautiful cloud formations behind aircraft. shot at gun butts.

257009. AV-8B night attack BU# 162966 aircraft (tail 162966 and nose 87 with night hawk logo) from MAD (Marine Air Detachment). sunset and special lighting. full side view.

257019. AV-8B night attack BU# 162966 aircraft (tail 162966 and nose 87 with night hawk logo) from MAD (Marine Air Detachment). sunset and special lighting. closeup side view of nose and cockpit.

257055. graphics composite of F/A-18A aircraft firing Sidewinder AIM 9R. aircraft piloted by Cdr

Benjes. intercept with QF-86 drone on April 12, 1989.

257110. VX-5 A-6E BU# 161084 night attack aircraft (nose 22 and XE on tail) preparing for takeoff. sunset over the Sierras. pilot Lt Rzeszotko and back seat Lt Weston. excellent 3/4 front view of aircraft and cloud formations.

257135. VX-5 F/A-18A night attack aircraft (nose 34 and tail 34). pilot Cdr Antonio with night vision goggles. close port side view of pilot in cockpit.

257274. air to air view of four plane formation. clockwise from left to right: VX-5"s EA-6B, NWC"s A-7E, NWC"s A-6E, and NWC"s F/A-18A. all aircraft loaded with AGM-88 Harm. overhead right side view.

258795. VX-5 F/A-18A aircraft (nose 35) uploaded with Harm, Maverick, and Sidewinder missiles. Cdr Antonio with night vision goggles in cockpit. full side view of aircraft just after sunset.

262865. 3/4 front view of Sidewinder AIM 9R missile on stand.

262866. front view of Sidewinder AIM 9R missile on stand.

262867. 3/4 front view of Sidewinder AIM 9R missile on stand.

262868. Sidewinder AIM 9R missile mounted on F/A-18C BU# 163284 aircraft (nose 110). closeup side view of missile on outboard wing pylon.

262869. Sidewinder AIM 9R missile mounted on F/A-18C BU# 163284 aircraft (nose 110). closeup front view of missile on outboard wing pylon.

262870. Sidewinder AIM 9R missile mounted on F/A-18C BU# 163284 aircraft (nose 110). closeup view of front of missile and launcher.

262871. Sidewinder AIM 9R missile mounted on F/A-18C BU# 163284 aircraft (nose 110). 3/4 front view of aircraft.

262872. Sidewinder AIM 9R missile mounted on F/A-18C BU# 163284 aircraft (nose 110). 3/4 front view of aircraft.

262873. Sidewinder AIM 9R missile mounted on F/A-18C BU# 163284 aircraft (nose 110). full side view of aircraft.

264968. F/A-18 firing Sidewinder AIM 9R. missile just intercepting QF-86 drone.

900304. air to air view of Northrop ATB (Advanced Technology Stealth Bomber) B-2 aircraft. first public flight in late 1989.

APPENDIX C

LEXICON

```
adj('FMC', fp(char)).
adj('in charge', fp(event)).
adj('in command', fp(event)).
adj('in contact', fp(event)).
adj('in effect', fp(event)).
adj('in line', fp(event)).
adj('in open', fp(event)).
adj('in place', fp(event)).
adj('in the open', fp(event)).
adj('in touch with', fp(event)).
adj('on the move', fp(event)).
adj('out of the way', fp(event)).
adj('squared away', fp(event)).
adj(able, fp(char)).
adj(accurate, fp(quality)).
adj(actual, fp(char)).
adj(aft, fp(location), fpcat(rear)).
adj(airborne, fp(location), fpcat(air)).
adj(alert, fp(event), fpcat(warn)).
adj(alive, fp(char)).
adj(annular, fp(geometry), fpcat(round)).
adj(approximate, fp(quantity)).
adj(armored, fp(event), fpcat(armor)).
adj(attached, fp(event), fpcat(assemble)).
adj(available, fp(char)).
adj(avionic, fp(char)).
adj(aware, fp(char)).
adj(back, fp(location)).
adj(bad, fp(quality)).
adj(badly, fp(quality), fpcat(bad)).
adj(beautiful, fp(quality)).
adj(best, fp(quality)).
adj(better, fp(quality)).
adj(big, fp(size), fpcat(large)).
adj(binary, fp(quantity)).
adj(bright, fp(quality)).
adj(broke, fp(event), fpcat(break)).
adj(broken, fp(event), fpcat(break)).
adj(burned, fp(event), fpcat(burn)).
adj(burning, fp(event), fpcat(burn)).
adj(busy, fp(char)).
adj(camouflaged, fp(event), fpcat(camouflage)).
adj(capped, fp(event), fpcat(include)).
adj(careful, fp(char)).
adj(centerline, fp(position)).
adj(clear, fp(event), fpcat(free)).
adj(clockwise, fp(direction)).
adj(close, fp(event)).
adj(closed, fp(event), fpcat(close)).
adj(closer, fp(event), fpcat(close)).
adj(closest, fp(event), fpcat(close)).
adj(cold, fp(quality)).
adj(complete, fp(event)).
adj(composite, fp(char)).
```

```
adj(correct, fp(event)).
adj(covered, fp(event), fpcat(include)).
adj(critical, fp(char)).
adj(current, fp(time)).
adj(dark, fp(quality)).
adj(dead, fp(event)).
adj(destroyed, fp(event), fpcat(destroy)).
adj(diamond, fp(geometry)).
adj(different, fp(char)).
adj(digital, fp(char)).
adj(direct, fp(event)).
adj(dismounted, fp(event), fpcat(dismount)).
adj(distorted, fp(event), fpcat(distort)).
adj(double, fp(quantity), fpcat(2)).
adj(down, fp(direction)).
adj(earlier, fp(time)).
adj(early, fp(time)).
adj(easier, fp(char)).
adj(easiest, fp(char)).
adj(easy, fp(char)).
adj(electric, fp(char)).
adj(electrical, fp(char), fpcat(electric)).
adj(enough, fp(quantity)).
adj(exact, fp(quantity)).
adj(excellent, fp(char)).
adj(extended, fp(event), fpcat(extend)).
adj(exterior, fp(location), fpcat(outside)).
adj(extreme, fp(quantity)).
adj(far, fp(distance)).
adj(farther, fp(distance)).
adj(fast, fp(speed)).
adj(faster, fp(speed)).
adj(fastest, fp(speed)).
adj(final, fp(position)).
adj(fine, fp(quality)).
adj(flashing, fp(event), fpcat(flash)).
adj(folded, fp(event), fpcat(fold)).
adj(forward, fp(position), fpcat(front)).
adj(friendly, fp(char), fpcat(friend)).
adj(front, fp(position)).
adj(fwd, fp(position), fpcat(front)).
adj(full, fp(size)).
adj(garbled, fp(event), fpcat(garble)).
adj(good, fp(quality)).
adj(graphics, fp(char)).
adj(gutted, fp(event), fpcat(destroy)).
adj(hard, fp(quality)).
adj(hasty, fp(time)).
adj(heavy, fp(weight)).
adj(high, fp(height)).
adj(higher, fp(height)).
adj(highest, fp(height)).
adj(historical, fp(char), fpcat(history)).
adj(hot, fp(quality)).
adj(hydraulic, fp(char)).
adj(ignited, fp(event), fpcat(ignite)).
adj(immediate, fp(time)).
adj(important, fp(char)).
adj(inboard, fp(location)).
adj(inert, fp(char)).
adj(inside, fp(location)).
```

```
adj(interior, fp(location), fpcat(inside)).
adj(key, fp(char)).
adj(large, fp(size)).
adj(last, fp(position)).
adj(late, fp(time)).
adj(later, fp(time)).
adj(lateral, fp(position)).
adj(least, fp(quantity)).
adj(left, fp(direction)).
adj(less, fp(quantity)).
adj(level, fp(position)).
adj(light, fp(quantity)).
adj(little, fp(quantity)).
adj(local, fp(location)).
adj(long, fp(length)).
adj(longer, fp(length)).
adj(longest, fp(length)).
adj(loose, fp(char)).
adj(low, fp(height)).
adj(main, fp(char)).
adj(medium, fp(char)).
adj(mod, fp(event), fpcat(modify)).
adj(modified, fp(event), fpcat(modify)).
adj(more, fp(quantity)).
adj(most, fp(quantity)).
adj(much, fp(quantity)).
adj(naval, fp(char), fpcat(navy)).
adj(near, fp(position)).
adj(negative, fp(char)).
adj(neutralized, fp(event), fpcat(neutralize)).
adj(new, fp(quality)).
adj(next, fp(position)).
adj(nuclear, fp(char)).
adj(old, fp(quality)).
adj(only, fp(quantity)).
adj(open, fp(event), fpcat(open_act)).
adj(operational, fp(event), fpcat(perform)).
adj(opposite, fp(position)).
adj(out, fp(location)).
adj(outboard, fp(location)).
adj(outdoor, fp(location), fpcat(outside)).
adj(outside, fp(location)).
adj(overhead, fp(location)).
adj(own, fp(char)).
adj(photographic, fp(char), fpcat(photograph)).
adj(pictorial, fp(char), fpcat(picture)).
adj(port, fp(position), fpcat(left)).
adj(positive, fp(char)).
adj(possible, fp(char)).
adj(post, fp(time)).
adj(posttest, fp(time)).
adj(potted, fp(event), fpcat(pot)).
adj(powered, fp(event), fpcat(power)).
adj(preflight, fp(time)).
adj(prepared, fp(event), fpcat(prepare)).
adj(present, fp(time)).
adj(pretest, fp(time)).
adj(pretty, fp(quality)).
adj(probable, fp(char)).
adj(public, fp(quality)).
adj(readable, fp(event), fpcat(read)).
```

```
adj(ready, fp(time)).
adj(real, fp(quality)).
adj(rear, fp(position)).
adj(receding, fp(event), fpcat(depart)).
adj(regular, fp(time)).
adj(reverse, fp(position)).
adj(right, fp(direction)).
adj(safe, fp(char)).
adj(same, fp(char)).
adj(secure, fp(event)).
adj(sectioned, fp(event), fpcat(section)).
adj(shaped, fp(event), fpcat(shape)).
adj(short, fp(height)).
adj(slight, fp(char)).
adj(slow, fp(speed)).
adj(small, fp(size)).
adj(smouldering, fp(event), fpcat(smoulder)).
adj(solar, fp(char)).
adj(special, fp(char)).
adj(specific, fp(char)).
adj(square, fp(geometry)).
adj(stationary, fp(char)).
adj(starboard, fp(direction), fpcat(right)).
adj(stbd, fp(direction), fpcat(right)).
adj(straight, fp(position)).
adj(strange, fp(char)).
adj(stunned, fp(event), fpcat(shock)).
adj(sure, fp(char)).
adj(synchro, fp(char), fpcat(synchronous)).
adj(tight, fp(char)).
adj(unable, fp(char)).
adj(unfixable, fp(char)).
adj(unfilled, fp(event), fpcat(free)).
adj(unknown, fp(char)).
adj(unpersistent, fp(char)).
adj(unreadable, fp(char)).
adj(unstuck, fp(event)).
adj(vertical, fp(position)).
adj(visible, fp(char)).
adj(visual, fp(char)).
adj(weak, fp(quality)).
adj(whole, fp(quantity)).
adj(wild, fp(quality)).
adj(wrong, fp(char)).

adv('a bit', fp(quantity)).
adv('a little bit', fp(quantity)).
adv('a little while', fp(time)).
adv('a little', fp(quantity)).
adv('at all', fp(quantity)).
adv('at least', fp(quantity)).
adv('back and forth', fp(direction)).
adv('dead ahead', fp(position)).
adv('for a little bit', fp(quantity)).
adv('for a little while', fp(time)).
adv('for awhile', fp(time)).
%adv('kind of', fp(char)).
adv('on top', fp(location)).
adv('out of range', fp(position)).
adv('point blank', fp(position)).
adv('quite awhile', fp(time)).
```

```
adv('so forth', fp(char)).
adv('up here', fp(location)).
adv('up there', fp(location)).
adv(about, fp(char)).
adv(abreast, fp(position)).
adv(across, fp(location)).
adv(again, fp(quantity)).
adv(ago, fp(time)).
adv(ah, fp(char)).
adv(ahead, fp(position)).
adv(almost, fp(char)).
adv(already, fp(time)).
adv(also, fp(char)).
adv(always, fp(quantity)).
adv(anyway, fp(char)).
adv(apparently, fp(char)).
adv(approx, fp(quantity)).
adv(approximately, fp(quantity)).
adv(around, fp(direction)).
adv(away, fp(position)).
adv(awhile, fp(time)).
adv(back, fp(location)).
adv(before, fp(time)).
adv(behind, fp(position)).
adv(below, fp(position)).
adv(by, fp(char)).
adv(carefully, fp(quality)).
adv(clear, fp(quality)).
adv(clearly, fp(quality)).
adv(currently, fp(time)).
adv(definitely, fp(char)).
adv(directly, fp(char)).
adv(down, fp(direction)).
adv(due, fp(time)).
adv(easily, fp(quality)).
adv(enroute, fp(location)).
adv(even, fp(quality)).
adv(ever, fp(time)).
adv(exactly, fp(quantity)).
adv(far, fp(distance)).
adv(finally, fp(time)).
adv(forward, fp(direction), fpcat(front)).
adv(forwards, fp(direction), fpcat(front)).
adv(fully, fp(quantity)).
adv(further, fp(quantity)).
adv(hard, fp(quality)).
adv(hopefully, fp(char)).
adv(immediately, fp(time)).
adv(in, fp(location)).
adv(inside, fp(location)).
adv(instead, fp(char)).
adv(internally, fp(location)).
adv(just, fp(char)).
adv(maybe, fp(char)).
adv(nearly, fp(char)).
adv(off, fp(status)).
adv(often, fp(time)).
%adv(on, fp(status)).        % can't remember which captions bombed because of this
adv(once, fp(quantity)).
adv(only, fp(quantity)).
adv(out, fp(location)).
```

```
adv(outside, fp(location)).
adv(parallel, fp(position)).
adv(please, fp(char)).
adv(possibly, fp(char)).
adv(pretty, fp(quality)).
adv(prior, fp(time)).
adv(probably, fp(char)).
adv(quick, fp(time)).
adv(quickly, fp(time), fpcat(quick)).
adv(quite, fp(char)).
adv(real, fp(quality)).
adv(really, fp(char)).
adv(right, fp(direction)).
adv(set, fp(char)).
adv(shortly, fp(time)).
adv(slowly, fp(speed)).
adv(so, fp(char)).
adv(sometime, fp(time)).
adv(sometimes, fp(char)).
adv(somewhere, fp(location)).
adv(soon, fp(time)).
adv(sooner, fp(time)).
adv(still, fp(char)).
adv(straight, fp(quality)).
adv(through, fp(char)).
adv(together, fp(char)).
adv(too, fp(char)).
adv(towards, fp(direction)).
adv(ultimately, fp(char)).
adv(under, fp(location)).
adv(underway, fp(char)).
adv(up, fp(direction)).
adv(very, fp(char)).
adv(visually, fp(char)).
adv(well, fp(char)).
adv(worse, fp(quality)).
adv(yet, fp(time)).

art(a).
art(an).
art(the).

be('''m').
be('''re').
be('''s').
be('ai').
be(am).
be(are).
be(be).
be(been).
be(being).
be(is).
be(was).
be(were).

%bprep(with).

by(by).
by(x).

cardinal(eight, fpcat(8)).
```

```
cardinal(eleven, fpcat(11)).
cardinal(five, fpcat(5)).
cardinal(four, fpcat(4)).
cardinal(nine, fpcat(9)).
cardinal(one, fpcat(1)).
cardinal(seven, fpcat(7)).
cardinal(six, fpcat(6)).
cardinal(ten, fpcat(10)).
cardinal(three, fpcat(3)).
cardinal(twelve, fpcat(12)).
cardinal(two, fpcat(2)).
cardinal(hundred, fpcat(100)).

colon(':').

comma(',').

comp(that).

conj(and).
conj(but).
conj(nor).
conj(or).

correc(correction).

demadv(here).
demadv(then).
demadv(there).

dempro(that).
dempro(these).
dempro(this).
dempro(those).

dir('due east', fp(direction)).
dir('due north', fp(direction)).
dir('due south', fp(direction)).
dir('due west', fp(direction)).
dir('hard left', fp(direction)).
dir('hard right', fp(direction)).
dir('left hand', fp(direction)).
dir('right hand', fp(direction)).
dir(down, fp(direction)).
dir('E', fp(direction), fpcat(east)).
dir(east, fp(direction)).
dir(left, fp(direction)).
dir('N', fp(direction), fpcat(north)).
dir(north, fp(direction)).
dir(northeast, fp(direction)).
dir(northwest, fp(direction)).
dir(right, fp(direction)).
dir('S', fp(direction), fpcat(south)).
dir(south, fp(direction)).
dir(southeast, fp(direction)).
dir(southwest, fp(direction)).
dir(up, fp(direction)).
dir('W', fp(direction), fpcat(west)).
dir(west, fp(direction)).

diradj('left hand', fp(position)).
```

```
diradj('right hand', fp(position)).
diradj(east, fp(position)).
diradj(easterly, fp(position)).
diradj(eastern, fp(position)).
diradj(left, fp(position)).
diradj(north, fp(position)).
diradj(northeast, fp(position)).
diradj(northerly, fp(position)).
diradj(northern, fp(position)).
diradj(northwest, fp(position)).
diradj(northwesterly, fp(position)).
diradj('NW', fp(position), fpcat(northwest)).
diradj(right, fp(position)).
diradj(south, fp(position)).
diradj(southeast, fp(position)).
diradj(southeasterly, fp(position)).
diradj(southerly, fp(position)).
diradj(southern, fp(position)).
diradj(southwest, fp(position)).
diradj(west, fp(position)).
diradj(westerly, fp(position)).
diradj(western, fp(position)).

do(did).
do(do).
do(does).

everpro(however,fp(adv)).
everpro(whatever,fp(noun)).
everpro(whenever,fp(adv)).
everpro(wherever,fp(adv)).
everpro(whichever,fp(adj)).
everpro(whoever,fp(noun)).

exp('Negative sightings').
exp('action left').
exp('action right').
exp('check it out').
exp('come on').
exp('contact left').
exp('contact right').
exp('continue to observe').
exp('don''t worry').
exp('enemy front').
exp('enemy left').
exp('enemy right').
exp('heads up').
exp('hold on').
exp('no telling').
exp('request permission to engage').
exp('request permission to fire').
exp('there''s no way').
exp('work it out').

expro(there).

gen('''s').

gon('going to').
gon(gonna).
```

```
got('got to').
got(gotta).

have('''s').
have('''ve').
have(has).
have(have).

infin('hold fast', morph(1-b), irforms(held), fpcat(inhabit)).
infin('off-load', morph(1-a), fpcat(disassemble)).
infin('on fire', morph(5-c), fpcat(heat)).
infin(accomplish, morph(2-a), fpcat(complete)).
infin(account, morph(1-a), case([[vprep(for), dobj(direct_object)]])).
infin(add, morph(1-a), fpcat(increase)).
infin(adjust, morph(1-a), fpcat(fix)).
infin(advise, morph(3-a), fpcat(mtrans)).
infin(agree, morph(1-b), irforms(agreed)).
infin(aid, morph(1-a)).
infin(alert, morph(1-a), fpcat(attend)).
infin(allow, morph(1-a), fpcat(permit)).
infin(answer, morph(1-a), fpcat(mtrans)).
infin(anticipate, morph(3-a), fpcat(expect)).
infin(appear, morph(1-a)).
infin(approach, morph(2-a),
    case([[dobj(phys_obj), vcase(destination)]])).
infin(arm, morph(1-a), fpcat(assemble),
    case([[vprep(with), vtype(assemble), vcase(object)]])).
infin(arrive, morph(3-a)).
infin(ask, morph(1-a), fpcat(inquire)).
infin(assemble, morph(3-a), case([[vprep(on)]])).
infin(assess, morph(2-a), fpcat(mbuild)).
infin(assign, morph(1-a), fpcat(atrans)).
infin(assist, morph(1-a), fpcat(aid)).
infin(assume, morph(3-a), fpcat(mbuild)).
infin(attach, morph(1-a), fpcat(assemble)).
infin(attack, morph(1-a)).
infin(attempt, morph(1-a), fpcat(try)).
infin(attend, morph(1-a)).
infin(avoid, morph(1-a)).
infin(back, morph(1-a), fpcat(ptrans),
    case([[vpart(off), vtype(ptrans)],
          [vpart(up), vtype(ptrans)]])).
infin(bank, morph(1-a), fpcat(ptrans)).
infin(bear, morph(1-a), fpcat(ptrans)).
infin(begin, morph(5-b), irforms(began,begun)).
infin(believe, morph(3-a), fpcat(mbuild)).
infin(belong, morph(1-a), fpcat(include)).
infin(bend, morph(1-b), irforms(bent), fpcat(curve)).
infin(blow, morph(1-b), irforms(blew,blown)).
infin(bound, morph(1-a), fpcat(ptrans)).
infin(break, morph(1-b), irforms(broke, broken), fpcat(injure),
    case([[dobj(phys_obj)]])).
infin(brief, morph(1-a), fpcat(mtrans)).
infin(bring, morph(1-b), irforms(brought), fpcat(ptrans),
    case([[vpart(back), dobj(phys_obj), iobj(phys_obj)]])).
infin(bump, morph(1-a), fpcat(hit),
    case([[dobj(phys_obj)],
          [vprep(into), dobj(phys_obj)]])).
infin(burn, morph(1-b), irforms(burned, burnt), fpcat(heat)).
infin(button, morph(1-a), fpcat(assemble), case([[vpart(up)]])).
infin(bypass, morph(2-a), fpcat(avoid)).
```

```
infin(call, morph(1-a), fpcat(mtrans)).
infin(cancel, morph(5-a), fpcat(cease)).
infin(carry, morph(4-a),
    case([[dobj(phys_obj), vcase(theme)]])).
infin(catch, morph(2-b), irforms(caught), fpcat(atrans)).
infin(cause, morph(3-a), fpcat(influence)).
infin(cease, morph(3-a)).
infin(change, morph(3-a)).
infin(charge, morph(3-a), fpcat(attack)).
infin(choose, morph(1-b), irforms(chose, chosen)).
infin(clear, morph(1-a), fpcat(free),
    case([[dobj(phys_obj), vcase(object)],
          [vprep(of), vtype(free), vcase(object)]])).
infin(close, morph(3-a)).
infin(collapse, morph(3-a), fpcat(fatigue)).
infin(color, morph(1-a)).
infin(come, morph(3-b), irforms(came,come), fpcat(arrive),
    case([[vprep(into), vtype(approach), vcase(destination)],
          [vprep(on)],
          [vpart(off), vtype(ptrans)],
          [vtype(ptrans)]])).
infin(compare, morph(3-a)).
infin(complete, morph(3-a)).
infin(conceal, morph(1-a)).
infin(conceive, morph(3-a)).
infin(connect, morph(1-a), fpcat(assemble)).
infin(contact, morph(1-a), fpcat(arrive)).
infin(continue, morph(3-a), case([[dobj(infinphr), vtype(aspect)],
[dobj(phys_obj)]])).
infin(control, morph(5-a), fpcat(influence)).
infin(copy, morph(4-a), fpcat(imitate)).
infin(correct, morph(1-a), fpcat(agree)).
infin(count, morph(1-a), fpcat(number)).
infin(cover, morph(1-a), fpcat(include),
    case([[vprep(with), vtype(include), vcase(object)]])).
infin(cross, morph(1-a), fpcat(ptrans)).
infin(cruise, morph(3-a), fpcat(ptrans)).
infin(curve, morph(3-a)).
infin(cut, morph(5-b), irforms(cut), fpcat(divide)).
infin(damage, morph(3-a), fpcat(injure)).
infin(debrief, morph(1-a), fpcat(mtrans)).
infin(decide, morph(3-a), fpcat(choose)).
infin(decrease, morph(3-a)).
infin(delay, morph(1-a), fpcat(wait)).
infin(deliver, morph(1-a), fpcat(ptrans)).
infin(depart, morph(1-a)).
infin(depend, morph(1-a)).
infin(destroy, morph(1-a), fpcat(injure), case([[dobj(phys_obj)]])).
infin(direct, morph(1-a)).
infin(disassemble, morph(3-a)).
infin(discover, morph(1-a)).
infin(discuss, morph(2-a), fpcat(mtrans)).
infin(dismount, morph(1-a), fpcat(disassemble), case([[vtype(changepos)]])).
infin(disregard, morph(1-a), fpcat(avoid)).
infin(divide, morph(3-a)).
infin(do, morph(2-b), irforms(did,done), fpcat(perform)).
infin(doubt, morph(1-a)).
infin(drive, morph(3-b), irforms(drove,driven), fpcat(propel)).
infin(drop, morph(5-a), fpcat(launch)).
infin(dump, morph(1-a), fpcat(disassemble),
    case([[vprep(from), vtype(disassemble), vcase(source)]])).
```

```
infin(eat, morph(1-b), irforms(ate,eaten), fpcat(ingest)).
infin(educate, morph(3-a)).
infin(eject, morph(1-a), fpcat(expel)).
infin(emerge, morph(3-a), fpcat(expel)).
infin(empty, morph(4-a), fpcat(free)).
infin(engage, morph(3-a), fpcat(attack), case([[dobj(participant),
actor(participant)]])).
infin(ensure, morph(3-a), fpcat(mbuild)).
infin(equip, morph(5-a), fpcat(assemble)).
infin(erode, morph(3-a), fpcat(decrease)).
infin(estimate, morph(3-a), fpcat(measure)).
infin(evaluate, morph(3-a), fpcat(inspect)).
infin(examine, morph(3-a), fpcat(inspect), case([[dobj(phys_obj)]])).
infin(exchange, morph(3-a), fpcat(atrans)).
infin(execute, morph(3-a), fpcat(perform)).
infin(expand, morph(1-a)).
infin(expect, morph(1-a)).
infin(expend, morph(1-a), fpcat(launch)).
infin(explode, morph(3-a)).
infin(extend, morph(1-a), fpcat(expand),
    case([[vpart(back), vtype(expand)]])).
infin(face, morph(3-a), fpcat(observe),
    case([[dobj(phys_obj)]])).
infin(fatigue, morph(3-a)).
infin(feed, morph(1-b), irforms(fed), fpcat(ingest),
    case([[idiomobj(chow)]])).
infin(fight, morph(1-b), irforms(fought),
    case([[dobj(phys_obj)]])).
infin(figure, morph(3-a), fpcat(measure)).
infin(find, morph(1-b), irforms(found), fpcat(discover),
    case([[dobj(phys_obj)]])).
infin(finish, morph(2-a), fpcat(complete),
    case([[dobj(phys_obj)]])).
infin(fire, morph(3-a), fpcat(launch),
    case([[vprep(at), vtype(launch), vcase(destination)],
          [dobj(phys_obj)]])).
infin(fix, morph(2-a)).
infin(flag, morph(5-a), fpcat(mtrans)).
infin(flash, morph(2-a), fpcat(mtrans)).
infin(fly, morph(4-b), irforms(flies, flew)).
infin(follow, morph(1-a), fpcat(pursue)).
infin(forget, morph(5-b), irforms(forgot, forgotten)).
infin(free, morph(3-a)).
infin(function, morph(1-a), fpcat(perform)).
infin(gaze, morph(3-a), fpcat(observe),
    case([[vprep(at), vtype(observe), vcase(theme)]])).
infin(generate, morph(3-a), fpcat(pbuild)).
infin(get, morph(5-b), irforms(got,gotten), cat(pastpart, got), fpcat(atrans),
    case([[dobj(phys_obj)],
          [vpart(out), dobj(phys_obj), vtype(changestate)],
          [idiomad(set), vtype(readiness)],
          [idiomad(ready)]])).
infin(give, morph(3-b), irforms(gave, given), fpcat(atrans),
    case([[idiomobj(order)]])).
infin(go, morph(2-b), irforms(went,gone), fpcat(ptrans),
    case([[vpart(back), vtype(ptrans)],
          [idiomad(ahead), vtype(continue)]])).
infin(guess, morph(2-a), fpcat(choose)).
infin(happen, morph(1-a), fpcat(occur)).
infin(haul, morph(1-a), fpcat(ptrans),
    case([[dobj(phys_obj)]])).
```

```
infin(have, morph(3-c), irforms(has,had,had), fpcat(atrans),
    case([[dobj(phys_obj)],
          [dobj(phys_obj), vobj(infin)]])).
infin(head, morph(1-a), fpcat(direct),
    case([[oobj(dir)]])).
infin(hear, morph(1-b), irforms(heard), fpcat(attend)).
infin(heat, morph(1-b), irforms(heated)).
infin(help, morph(1-a), fpcat(aid)).
infin(hide, morph(3-b), irforms(hid, hidden), fpcat(conceal)).
infin(hit, morph(5-b), irforms(hit),
    case([[dobj(phys_obj)]])).
infin(hoist, morph(1-a), fpcat(ptrans)).
infin(hold, morph(1-b), irforms(held), fpcat(assemble)).
infin(holler, morph(1-a), fpcat(speak)).
infin(hook, morph(1-a), fpcat(assemble)).
infin(hope, morph(3-a), fpcat(expect)).
infin(hurt, morph(1-b), irforms(hurt), fpcat(injure)).
infin(identify, morph(4-a), fpcat(discover)).
infin(imitate, morph(3-a)).
infin(impact, morph(1-a), fpcat(hit),
    case([[dobj(phys_obj)]])).
infin(include, morph(3-a)).
infin(increase, morph(3-a)).
infin(inform, morph(1-a), fpcat(mtrans)).
infin(influence, morph(3-a)).
infin(inhabit, morph(1-a)).
infin(initiate, morph(3-a), fpcat(begin)).
infin(injure, morph(3-a)).
infin(inquire, morph(3-a)).
infin(inspect, morph(1-a),
    case([[dobj(phys_obj)]])).
infin(install, morph(1-a), fpcat(assemble),
    case([[vneg(not), vtype(disassemble)]])).
infin(instruct, morph(1-a), fpcat(mbuild)).
infin(intercept, morph(1-a), fpcat(hit),
    case([[dobj(phys_obj)]])).
infin(join, morph(1-a), fpcat(assemble)).
infin(jump, morph(1), fpcat(ptrans)).
infin(keep, morph(1-b), irforms(kept), fpcat(atrans)).
infin(key, morph(1-a), fpcat(influence)).
infin(kill, morph(1-a), fpcat(injure)).
infin(know, morph(1-b), irforms(knew,known),
    case([[dobj(phys_obj), vtype(mbuild)]])).
infin(launch, morph(1-a)).
infin(lead, morph(1-b), fpcat(direct)).
infin(leave, morph(3-b), fpcat(depart),
    case([[dobj(phys_obj), vcase(source)]])).
infin(lend, morph(3-b), irforms(lended, lent), fpcat(atrans)).
infin(let, morph(5-b), irforms(let), fpcat(permit)).
infin(lift, morph(1-a), fpcat(ptrans),
    case([[vpart(off), vtype(depart)]])).
infin(link, morph(1-a), fpcat(assemble)).
infin(listen, morph(1-a), fpcat(attend)).
infin(load, morph(1-a), fpcat(assemble),
    case([[vprep(on)],
          [vprep(onto)]])).
infin(locate, morph(3-a), fpcat(discover)).
infin(lock, morph(1-a), fpcat(assemble)).
infin(look, morph(1-a), fpcat(observe),
    case([[vprep(for), dobj(phys_obj),vtype(action)],
          [dobj(direction)]])).
```

```
infin(lose, morph(3-b), irforms(lost), fpcat(forget)).
infin(make, morph(3-b), irforms(made), fpcat(pbuild),
    case([[idiomad(sure)]])).
infin(maneuver, morph(1-a), fpcat(ptrans)).
infin(mask, morph(1-a), fpcat(conceal)).
infin(measure, morph(3-a)).
infin(meet, morph(1-b), irforms(met), fpcat(arrive)).
infin(miss, morph(2-a), fpcat(avoid)).
infin(mount, morph(1-a), fpcat(assemble),
    case([[vprep(on)],
          [vtype(changepos)]])).
infin(move, morph(3-a), fpcat(ptrans),
    case([[vpart(out),vtype(ptrans)],
          [vpart(up),dobj(phys_obj)],
          [oobj(dir),vtype(ptrans)],
          [vtype(ptrans)]])).
infin(need, morph(1-a), fpcat(require)).
infin(note, morph(3-a), fpcat(observe)).
infin(notice, morph(3-a), fpcat(observe)).
infin(number, morph(1-a)).
infin(observe, morph(3-a), case([[dobj(phys_obj), vtype(observe)],
[vtype(observe)]])).
infin(obtain, morph(1-a), fpcat(atrans)).
infin(occupy, morph(4-a), fpcat(inhabit)).
infin(occur, morph(5-a)).
infin(open, morph(1-a), fpcat(open_act), case([[vprep(on)]])).
infin(operate, morph(3-a), fpcat(perform)).
infin(order, morph(1-a), fpcat(mtrans)).
infin(orient, morph(1-a), fpcat(direct), case([[voice(passive), oobj(dir)]])).
infin(outfit, morph(5-a), fpcat(assemble),
    case([[vprep(with), vtype(assemble), vcase(object)]])).
infin(paint, morph(1-a), fpcat(color)).
infin(park, morph(1-a), fpcat(inhabit)).
infin(pass, morph(2-a), fpcat(ptrans)).
infin(perform, morph(1-a)).
infin(permit, morph(5-a)).
infin(pilot, morph(1-a), fpcat(influence)).
infin(place, morph(3-a), fpcat(assemble),
    case([[dobj(phys_obj)]])).
infin(plan, morph(5-a), fpcat(mtrans)).
infin(point, morph(1-a), fpcat(direct)).
infin(pop, morph(5-a), fpcat(launch)).
%infin(position, morph(1-a), fpcat(assemble)).
infin(post, morph(1-a), fpcat(mtrans)).
infin(pour, morph(1-a), fpcat(ptrans),
    case([[vprep(from), vtype(depart), vcase(source)]])).
infin(prepare, morph(3-a),
    case([[vprep(for), vtype(prepare), vcase(goal)]])).
%   case([[vprep(to), vtype(prepare), vcase(accompaniment)]])).
infin(proceed, morph(1-a), fpcat(continue)).
infin(propel, morph(5-a)).
infin(provide, morph(3-a), fpcat(atrans)).
infin(pull, morph(1-a), fpcat(ptrans), case([[vpart(back), vtype(ptrans)],
[vpart(out), vtype(ptrans)]])).
infin(pursue, morph(3-a)).
infin(push, morph(2-a), fpcat(propel)).
infin(put, morph(5-b), irforms(put), fpcat(assemble)).
infin(quit, morph(5-b), irforms(quit), fpcat(complete)).
infin(reach, morph(2-a), fpcat(arrive),
    case([[dobj(phys_obj), vcase(destination)]])).
infin(read, morph(1-b), irforms(read), fpcat(mtrans)).
```

```
infin(realize, morph(3-a), fpcat(mbuild)).
infin(receive, morph(3-a), fpcat(atrans)).
infin(recognize, morph(3-a), fpcat(discover)).
infin(recon, morph(5-a), fpcat(discover)).
infin(recover, morph(1-a), fpcat(atrans)).
infin(relay, morph(1-a), fpcat(mtrans)).
infin(release, morph(3-a)).
infin(reload, morph(1-a), fpcat(assemble)).
infin(relocate, morph(3-a), fpcat(ptrans),
     case([[dobj(phys_obj)]])).
infin(remain, morph(1-a), fpcat(inhabit),
     case([[dobj(pastpart), vtype(aspect)]])).
infin(repair, morph(1-a), fpcat(fix)).
infin(repeat, morph(1-a), fpcat(imitate)).
infin(replace, morph(3-a), fpcat(fix)).
infin(reply, morph(4-a), fpcat(mtrans)).
infin(report, morph(1-a), fpcat(mtrans), case([[dobj(phys_obj),vtype(mtrans)]])).
infin(reposition, morph(1-a), fpcat(ptrans)).
infin(request, morph(1-a), case([[idiomobj(fire), actor(empty)]])).
infin(require, morph(3-a)).
infin(retreat, morph(1-a), fpcat(depart)).
infin(return, morph(1-a), fpcat(arrive)).
infin(ride, morph(3-a)).
infin(roll, morph(1-a), fpcat(ptrans)).
infin(run, morph(5-b), irforms(ran,run), fpcat(ptrans), case([[vtype(ptrans)]])).
infin(save, morph(3-a), fpcat(store)).
infin(say, morph(1-b), irforms(said), fpcat(speak), case([[idiomad(again),
vtype(speak)]])).
infin(scan, morph(5-a), fpcat(inspect)).
infin(secure, morph(3-a), fpcat(assemble)).
infin(see, morph(1-b), fpcat(observe), irforms(saw,seen),
case([[dobj(phys_obj),vtype(observe)]])).
infin(seize, morph(2-a), fpcat(atrans)).
infin(select, morph(1-a), fpcat(choose), case([[dobj(phys_obj)]])).
infin(send, morph(1-b), irforms(sent), fpcat(ptrans), case([[dobj(phys_obj)],
[dobj(phys_obj), iobj(phys_obj)], [dobj(phys_obj), vtype(atrans)]])).
infin(shift, morph(1-a), fpcat(ptrans)).
infin(shock, morph(1-a)).
infin(shoot, morph(1-b), irforms(shot), fpcat(launch),
case([[voice(passive),actor(empty)]])).
infin(show, morph(1-b), irforms(showed, shown)).
infin(shut, morph(5-b), irforms(shut), fpcat(close)).
infin(sight, morph(1-a), fpcat(observe)).
infin(sit, morph(5-b), irforms(sat), fpcat(inhabit), case([[vtype(inhabit)]])).
infin(smoulder, morph(1-a), fpcat(heat)).
infin(solder, morph(1-a),
     case([[dobj(phys_obj)]])).
infin(sound, morph(1-a), fpcat(speak)).
infin(speed, morph(1-b), irforms(sped), fpcat(ptrans)).
infin(spot, morph(5-a), fpcat(observe)).
infin(spread, morph(1-a)).
infin(speak, morph(1-b), irforms(spoke, spoken)).
infin(stand, morph(1-b), irforms(stood), fpcat(inhabit),
     case([[vpart(by), vtype(wait)]])).
infin(start, morph(1-a), fpcat(begin),
     case([[dobj(vnoun), vtype(begin)],
            [dobj(infinphr), vtype(begin)]])).
infin(stay, morph(1-a), fpcat(inhabit)).
infin(stick, morph(1-b), irforms(stuck), fpcat(assemble)).
infin(stop, morph(5-a), fpcat(cease), case([[vtype(cease)]])).
infin(store, morph(3-a)).
```

```
infin(stow, morph(1-a), fpcat(store)).
infin(submit, morph(5-a), fpcat(atrans)).
infin(supply, morph(4-a), fpcat(atrans)).
infin(support, morph(1-a), fpcat(perform)).
infin(survive, morph(3-a)).
infin(swap, morph(5-a), fpcat(switch)).
infin(swing, morph(1-a), fpcat(ptrans)).
infin(switch, morph(2-a)).
infin(take, morph(3-b), irforms(took, taken), fpcat(atrans),
    case([[vprep(off), vtype(depart)],
          [vpart(off), vtype(depart)]])).
infin(talk, morph(1-a), fpcat(speak)).
infin(tell, morph(1-b), irforms(told), fpcat(mtrans), case([[dobj(phys_obj),
vtype(mtrans)]])).
%infin(test, morph(1-a)).
infin(think, morph(1-b), irforms(thought), fpcat(mbuild)).
infin(tie, morph(1-b), irforms(tied), fpcat(assemble)).
infin(tighten, morph(1-a), fpcat(assemble)).
infin(touch, morph(1-a), fpcat(hit),
    case([[vpart(down), vtype(arrive)],
          [dobj(phys_obj)]])).
infin(tow, morph(1-a), fpcat(ptrans), case([[dobj(phys_obj)]])).
infin(track, morph(1-a), fpcat(pursue)).
%infin(train, morph(1-a), fpcat(educate)).
infin(travel, morph(1-a), fpcat(ptrans)).
infin(traverse, morph(3-a), fpcat(ptrans)).
infin(try, morph(4-a)).
infin(turn, morph(1-a), fpcat(ptrans),
    case([[vpart(on), vtype(perform)]])).
infin(understand, morph(1-b), irforms(understood), fpcat(mbuild),
    case([[dobj(phys_obj), vtype(mbuild)]])).
infin(unfill, morph(1-a), fpcat(free)).
infin(unload, morph(1-a), fpcat(disassemble), case([[vprep(off)]])).
infin(upload, morph(1-a), fpcat(assemble), case([[vprep(on)]])).
infin(use, morph(3-a)).
infin(validate, morph(3-a)).
infin(wait, morph(1-a), case([[dobj(temp)]])).
infin(walk, morph(1-a), fpcat(ptrans)).
infin(want, morph(1-a), fpcat(require), case([[dobj(phys_obj), oobj(loc)],
[dobj(phys_obj)], [dobj(phys_obj), vobj(infinphr)]])).
infin(warn, morph(1-a), fpcat(mtrans)).
infin(watch, morph(2-a), fpcat(observe)).
infin(wave, morph(3-a), fpcat(mtrans)).
infin(wear, morph(1-b), irforms(wore, worn)).
infin(withdraw, morph(1-b), irforms(withdrew,withdrawn), fpcat(depart)).
infin(work, morph(1-a), fpcat(perform)).
infin(wrap, morph(5-a), fpcat(conceal)).
infin(write, morph(3-b), irforms(wrote, written), fpcat(mtrans)).
infin(yell, morph(1-a), fpcat(speak)).

infin_nom(keep, discontinuous([eye]), fpcat(observe)).
infin_nom(make, discontinuous([way]), fpcat(move)).

infin_part(button, discontinuous([up]), fpcat(close)).
infin_part(check, discontinuous([out])).
infin_part(close, discontinuous([up]), fpcat(close)).
infin_part(drop, discontinuous([off])).
infin_part(figure, discontinuous([out])).
infin_part(give, discontinuous([up])).
infin_part(make, discontinuous([out]), fpcat(observe)).
infin_part(pick, discontinuous([up])).
```

```
infin_part(police, discontinuous([up])).
infin_part(send, discontinuous([in])).
infin_part(send, discontinuous([up])).
infin_part(speed, discontinuous([up])).
infin_part(square, discontinuous([away])).
infin_part(top, discontinuous([off])).
infin_part(tuck, discontinuous([in])).
infin_part(turn, discontinuous([in])).
infin_part(turn, discontinuous([off])).
infin_part(turn, discontinuous([on])).
infin_part(turn, discontinuous([over])).
infin_part(wake, discontinuous([up])).

infin_phrase(cease, discontinuous([fire]), fpcat(cease)).
infin_phrase(give, discontinuous([a, call]), fpcat(mtrans)).
infin_phrase(give, discontinuous([the, word]), fpcat(mtrans)).
infin_phrase(keep, discontinuous([posted]), fpcat(mtrans)).

interpro('how far',fp(adv)).
interpro('how many',fp(quant)).
interpro('what else', fp(noun)).
interpro('what kind of',fp(adj)).
interpro('what type of',fp(adj)).
interpro(how,fp(adv)).
interpro(what,fp(noun)).
interpro(when,fp(adv)).
interpro(where,fp(adv)).
interpro(which,fp(adj)).
interpro(who,fp(noun)).
interpro(why,fp(adv)).

letter(alfa).
letter(bravo).
letter(charlie).
letter(delta).
letter(echo).
letter(foxtrot).
letter(golf).
letter(hotel).
letter(india).
letter(juliett).
letter(kilo).
letter(lima).
letter(l).
letter(mike).
letter(m).
letter(november).
letter(n).
letter(oscar).
letter(o).
letter(papa).
letter(p).
letter(quebec).
letter(q).
letter(romeo).
letter(r).
letter(sierra).
letter(s).
letter(tango).
letter(uniform).
letter(victor).
```

```
letter(whiskey).
letter(xray).
letter(yankee).
letter(zulu).

mod('''ll').
mod('ought to').
mod(ca).
mod(can).
mod(could).
mod(may).
mod(might).
mod(must).
mod(should).
mod(will).
mod(wo).
mod(would).

month('Apr', fpcat(apr)).
month('April', fpcat(apr)).
month('Aug', fpcat(aug)).
month('August', fpcat(aug)).
month('Dec', fpcat(dec)).
month('December', fpcat(dec)).
month('Feb', fpcat(feb)).
month('Febuary', fpcat(feb)).
month('Jan', fpcat(jan)).
month('January', fpcat(jan)).
month('Jul', fpcat(jul)).
month('July', fpcat(jul)).
month('Jun', fpcat(jun)).
month('June', fpcat(jun)).
month('Mar', fpcat(mar)).
month('March', fpcat(mar)).
month('May', fpcat(may)).
month('Nov', fpcat(nov)).
month('November', fpcat(nov)).
month('Oct', fpcat(oct)).
month('October', fpcat(oct)).
month('Sep', fpcat(sep)).
month('September', fpcat(sep)).

neg('n''t').
neg(never).
neg(not).

noun('A-3', morph(4), fp(aircraft)).
noun('A-3B', morph(4), fp(aircraft)).
noun('A-4', morph(4), fp(aircraft)).
noun('A-4B', morph(4), fp(aircraft)).
noun('A-4C', morph(4), fp(aircraft)).
noun('A-4E', morph(4), fp(aircraft)).
noun('A-4M', morph(4), fp(aircraft)).
noun('A-5', morph(4), fp(aircraft)).
noun('A-5C', morph(4), fp(aircraft)).
noun('A-6', morph(4), fp(aircraft)).
noun('A-6A', morph(4), fp(aircraft)).
noun('A-6E', morph(4), fp(aircraft)).
noun('A-7', morph(4), fp(aircraft)).
noun('A-7B/E', morph(4), fp(aircraft)).
noun('A-7C', morph(4), fp(aircraft)).
```

```
noun('A-7E', morph(4), fp(aircraft)).
noun('a/c', morph(4), fp(phys_obj), fpcat(aircraft)).
noun('A/C Survivability', morph(9), fp(laboratory), fpcat('Aircraft Survivability
Laboratory')).
noun('AAR-47', morph(4), fp(equipment)).
noun('AC-130', morph(4), fp(aircraft)).
noun('AC-130A', morph(4), fp(aircraft)).
noun('ACIMD', morph(4), fp(missile)).
noun('Adam', morph(9), fp(pod)).
noun('Advanced Common Intercept Missile Demonstration', morph(9), fp(missile),
fpcat('ACIMD')).
noun('Advanced Technology Stealth Bomber', morph(9), fp(aircraft), fpcat('ATB')).
noun('Aero 51A', morph(4), fp(vehicle)).
noun('Agile-Quickturn', morph(9), fp(missile)).
noun('AGM 45', morph(4), fp(missile), fpcat('AGM-45')).
noun('AGM 65', morph(4), fp(missile), fpcat('AGM-65')).
noun('AGM 65C', morph(4), fp(missile), fpcat('AGM-65C')).
noun('AGM 65D', morph(4), fp(missile), fpcat('AGM-65D')).
noun('AGM 65E', morph(4), fp(missile), fpcat('AGM-65E')).
noun('AGM 88', morph(4), fp(missile), fpcat('AGM-88')).
noun('AGM 123', morph(4), fp(missile), fpcat('AGM-123')).
noun('AGM 123A', morph(4), fp(missile), fpcat('AGM-123A')).
noun('AGM-45', morph(4), fp(missile)).
noun('AGM-65', morph(4), fp(missile)).
noun('AGM-65C', morph(4), fp(missile)).
noun('AGM-65D', morph(4), fp(missile)).
noun('AGM-65E', morph(4), fp(missile)).
noun('AGM-88', morph(4), fp(missile)).
noun('AGM-88I', morph(4), fp(missile)).
noun('AGM-123', morph(4), fp(missile)).
noun('AGM-123A', morph(4), fp(missile)).
noun('AH-1', morph(4), fp(aircraft)).
noun('AH-1J', morph(4), fp(aircraft)).
noun('AH-1W', morph(4), fp(aircraft)).
noun('AIM 9', morph(4), fp(missile), fpcat('AIM-9')).
noun('AIM 9B', morph(4), fp(missile), fpcat('AIM-9B')).
noun('AIM 9C', morph(4), fp(missile), fpcat('AIM-9C')).
noun('AIM 9D', morph(4), fp(missile), fpcat('AIM-9D')).
noun('AIM 9L', morph(4), fp(missile), fpcat('AIM-9L')).
noun('AIM 9M', morph(4), fp(missile), fpcat('AIM-9M')).
noun('AIM 9R', morph(4), fp(missile), fpcat('AIM-9R')).
noun('AIM 54', morph(4), fp(missile), fpcat('AIM-54')).
noun('AIM 54C', morph(4), fp(missile), fpcat('AIM-54C')).
noun('AIM-9', morph(4), fp(missile)).
noun('AIM-9B', morph(4), fp(missile)).
noun('AIM-9C', morph(4), fp(missile)).
noun('AIM-9D', morph(4), fp(missile)).
noun('AIM-9L', morph(4), fp(missile)).
noun('AIM-9M', morph(4), fp(missile)).
noun('AIM-9R', morph(4), fp(missile)).
noun('AIM-54', morph(4), fp(missile)).
noun('AIM-54C', morph(4), fp(missile)).
noun('Air Force Reserve', morph(9), fp(organization), fpcat('AFR')).
noun('Air Force', morph(9), fp(organization), fpcat('USAF')).
noun('Aircraft Survivability', morph(9), fp(laboratory), fpcat('Aircraft Survivability
Laboratory')).
noun('air to air missile', morph(1), fp(missile), fpcat('air-to-air missile')).
noun('air to air view', morph(1), fp(event), fpcat('air-to-air view')).
noun('air to ground view', morph(1), fp(event), fpcat('air-to-ground view')).
noun('air-to-air missile', morph(1), fp(missile)).
noun('air-to-air view', morph(1), fp(event)).
```

```
noun('air-to-ground view', morph(1), fp(event)).
noun('aircraft carrier', morph(1), fp(ship)).
noun('AJ', morph(4), fp(id)).
noun('ALQ 119', morph(4), fp(pod), fpcat('ALQ-119')).
noun('ALQ-119', morph(4), fp(pod)).
noun('Antonio', morph(9), fp('human being')).
noun('Argus Range', morph(9), fp(location)).
noun('Armitage Field', morph(9), fp(location)).
noun('Armitage', morph(9), fp(location), fpcat('Armitage Field')).
noun('ASROC', morph(4), fp(phys_obj), fpcat('ASROC engine')).
noun('Assault Breaker', morph(9), fp(vehicle)).
noun('ATB', morph(4), fp(aircraft)).
noun('AV-8', morph(4), fp(aircraft)).
noun('AV-8A', morph(4), fp(aircraft)).
noun('AV-8B', morph(4), fp(aircraft)).
noun('B-1-B', morph(4), fp(target)).
noun('B-2', morph(4), fp(aircraft)).
noun('B-61', morph(4), fp(bomb)).
noun('B57-VFA-82A', morph(4), fp(pod)).
noun('back seat', morph(1), fp(phys_obj)).
noun('Ballistic Test Vehicle', morph(9), fp(vehicle), fpcat('BTV')).
noun('Bat', morph(9), fp(missile)).
noun('Beavertail cactus', morph(9), fp(plant)).
noun('Beling', morph(9), fp('human being')).
noun('Benjes', morph(9), fp('human being')).
noun('Bigeye', morph(9), fp(weapon)).
noun('BLU-80', morph(4), fp(weapon)).
noun('BLU-80/B', morph(4), fp(weapon)).
noun('bomb skid', morph(1), fp(stand)).
noun('bottom view', morph(1), fp(event)).
noun('BQM', morph(4), fp(aircraft)).
noun('BQM control', morph(4), fp(control)).
noun('BQM-34S', morph(4), fp(aircraft)).
noun('BTV', morph(4), fp(vehicle)).
noun('BU#', morph(4), fp(id), fpcat(bureau_no)).
noun('Bullpup', morph(9), fp(missile)).
noun('C-130', morph(4), fp(aircraft)).
noun('C-130A', morph(4), fp(aircraft)).
noun('camera pod', morph(1), fp(pod)).
noun('Canada', morph(9), fp(location)).
noun('Capt', morph(4), fp(rank), fpcat('Captain')).
noun('Captain', morph(4), fp(rank)).
noun('Captive Flight Test', morph(9), fp(event), fpcat('CFT')).
noun('Cdr', morph(4), fp(rank), fpcat('Commander')).
noun('centerline station', morph(1), fp(phys_obj)).
noun('CFT', morph(4), fp(event)).
noun('China Lake', morph(9), fp(location)).
noun('circuit board', morph(1), fp(phys_obj)).
noun('CL', morph(4), fp(id)).
noun('CLAM B', morph(4), fp(bomb)).
noun('cloud cover', morph(6), fp(phys_obj), fpcat(cloud)).
noun('Code', morph(9), fp(organization)).
noun('Cold Line', morph(9), fp(location)).
noun('Condor', morph(9), fp(missile)).
noun('control tower', morph(1), fp(tower), fpcat('airport tower')).
noun('cookoff test', morph(1), fp(event)).
noun('Coso Range', morph(9), fp(location)).
noun('Coso', morph(9), fp(location), fpcat('Coso Range')).
noun('Crippen', morph(9), fp('human being')).
noun('cruise missile', morph(1), fp(missile)).
noun('CSWS dispenser', morph(4), fp(phys_obj)).
```

```
noun('CVA-16', morph(4), fp(ship)).
noun('CVA-59', morph(4), fp(ship)).
noun('data link pod', morph(1), fp(pod)).
noun('Dalea', morph(9), fp(plant)).
noun('DC-130', morph(4), fp(aircraft), fpcat('C-130')).
noun('DC-130A', morph(4), fp(aircraft), fpcat('C-130A')).
noun('DD-851', morph(4), fp(ship)).
noun('Dev-Assist', morph(4), fp(equipment)).
noun('disassembly', morph(6), fp(event)).
noun('DODX', morph(4), fp(vehicle)).
noun('DSU-28/B', morph(4), fp(equipment), fpcat('DSU-28B')).
noun('dump test', morph(1), fp(event)).
noun('DVT-2R', morph(4), fp(system)).
noun('DVT-7', morph(4), fp(system)).
noun('EA-6B', morph(4), fp(aircraft), fpcat('A-6B')).
noun('Earth', morph(9), fp(location)).
noun('ECM pod', morph(9), fp(pod)).
noun('EDM-4', morph(4), fp(engine)).
noun('Eglin AFB', morph(4), fp(organization)).
noun('Eglin Air Force Base', morph(9), fp(organization), fpcat('Eglin AFB')).
noun('electro optical device', morph(1), fp(equipment), fpcat('electro-optical
device')).
noun('electro-optical device', morph(1), fp(equipment)).
noun('escape system', morph(1), fp(system)).
noun('ESS', morph(4), fp(simulator)).
noun('ET-1', morph(4), fp('#')).
noun('ET-2', morph(4), fp('#')).
noun('EX-62', morph(4), fp(equipment)).
noun('EXT-024', morph(4), fp(test)).
noun('Expendable Seeker Simulator', morph(9), fp(simulator), fpcat('ESS')).
noun('explosive device', morph(1), fp(phys_obj)).
noun('F-3', morph(4), fp(aircraft)).
noun('F-3D', morph(4), fp(aircraft)).
noun('F-3D-1', morph(4), fp(aircraft), fpcat('F-3D-1')).
noun('F-3H', morph(4), fp(aircraft)).
noun('F-4', morph(4), fp(aircraft)).
noun('F-4B', morph(4), fp(aircraft)).
noun('F-4G', morph(4), fp(aircraft)).
noun('F-4J', morph(4), fp(aircraft), fpcat('F-4J')).
noun('F-9', morph(4), fp(aircraft)).
noun('F-14', morph(4), fp(aircraft)).
noun('F-14A', morph(4), fp(aircraft)).
noun('F-15', morph(4), fp(aircraft)).
noun('F-18', morph(4), fp(aircraft), fpcat('F/A-18')).
noun('F/A-18', morph(4), fp(aircraft)).
noun('F/A-18 simulator', morph(9), fp(simulator)).
noun('F/A-18A simulator', morph(9), fp(simulator)).
noun('F/A-18A Validation Lab', morph(9), fp(room), fpcat('F/A-18A Validation
Laboratory')).
noun('F/A-18A', morph(4), fp(aircraft)).
noun('F/A-18C', morph(4), fp(aircraft)).
noun('F3', morph(4), fp(aircraft), fpcat('F-3')).
noun('F3D-1', morph(4), fp(aircraft), fpcat('F-3D-1')).
noun('F3H', morph(4), fp(aircraft), fpcat('F-3H')).
noun('F3H-1', morph(4), fp(aircraft), fpcat('F-3H-1')).
noun('F4', morph(4), fp(aircraft), fpcat('F-4')).
noun('F4G', morph(4), fp(aircraft), fpcat('F-4G')).
noun('FAE', morph(4), fp(bomb)).
noun('Fat Albert', morph(8), fp(bomb), fpcat('Walleye')).
noun('FBM', morph(4), fp(missile)).
noun('feet per second', morph(4), fp(measure), fpcat(fps)).
```

```
noun('fiber optic', morph(1), fp(wire)).
noun('Fireye', morph(9), fp(bomb)).
noun('firing test', morph(1), fp(event), fpcat('test firing')).
noun('Fl', morph(4), fp(location), fpcat('Florida')).
noun('Fleet Ballistic Missile', morph(9), fp(missile), fpcat('FBM')).
noun('flight deck', morph(1), fp(location)).
noun('flight test', morph(1), fp(event)).
noun('FLIR', morph(4), fp(instrument), fpcat('FLIR detector')).
noun('flying eagle', morph(9), fp(marking)).
noun('flyover test', morph(1), fp(event)).
noun('forward part', morph(1), fp(position), fpcat(front)).
noun('frame view', morph(1), fp(event)).
noun('front view', morph(1), fp(event)).
noun('FSQ-12', morph(4), fp(engine)).
noun('FTV-2', morph(4), fp(missile)).
noun('Fulmer', morph(9), fp('human being')).
noun('fwd part', morph(1), fp(position), fpcat(front)).
noun('G Range', morph(9), fp(location)).
noun('G-1 Range', morph(9), fp(location)).
noun('G-1', morph(4), fp(location), fpcat('G-1 Range')).
noun('G-2 Range', morph(9), fp(location)).
noun('G-R simulator', morph(1), fp(phys_obj)).
noun('Gallinetti', morph(9), fp('human being')).
noun('Gatling gun', morph(1), fp(gun)).
noun('GBU-22', morph(4), fp(bomb)).
noun('GBU-22/B', morph(4), fp(bomb), fpcat('GBU-22B')).
noun('Germain', morph(9), fp('human being')).
noun('Goshawk', morph(9), fp(aircraft)).
noun('Guardian of the Skies', morph(9), fp(missile), fpcat('Sparrow 3')).
noun('guided missile submarine', morph(1), fp(ship)).
noun('Gulf of Tonkin', morph(9), fp(location)).
noun('H-6', morph(4), fp(explosive), fpcat('HBX')).
noun('hangar bay', morph(1), fp(location)).
noun('Harm', morph(9), fp(missile)).
noun('Harpoon', morph(9), fp(missile)).
noun('Harrier II', morph(9), fp(aircraft), fpcat('Harrier')).
noun('Harrier', morph(9), fp(aircraft)).
noun('Harvey Field', morph(9), fp(location)).
noun('HBX', morph(4), fp('#')).
noun('head on view', morph(1), fp(event), fpcat('front view')).
noun('Helicopter Trap Weapon', morph(9), fp(weapon), fpcat('HTW')).
noun('Hessler', morph(9), fp('human being')).
noun('HH-1', morph(4), fp(aircraft)).
noun('HH-1K', morph(4), fp(aircraft)).
noun('Hipeg', morph(9), fp(gun)).
noun('Honeywell', morph(9), fp(organization)).
noun('Hot Line', morph(9), fp(location)).
noun('HS camera', morph(1), fp(equipment)).
noun('HTW', morph(4), fp(weapon)).
noun('Hytas', morph(9), fp(radar)).
noun('IC', morph(4), fp(phys_obj), fpcat('integrated circuit')).
noun('inside view', morph(1), fp(event)).
noun('instrumentation pod', morph(1), fp(pod)).
noun('integrated circuit', morph(1), fp(phys_obj)).
noun('IR', morph(4), fp(instrument), fpcat('infrared detector')).
noun('IRAH', morph(4), fp(instrument)).
noun('ITER', morph(1), fp(launcher)).
noun('J-52', morph(4), fp(engine)).
noun('J52', morph(4), fp(engine), fpcat('J-52')).
noun('John', morph(9), fp('human being')).
noun('Jon', morph(9), fp('human being')).
```

```
noun('Kapernick', morph(9), fp('human being')).
noun('Kennedy', morph(9), fp('human being')).
noun('Kern River', morph(9), fp(location)).
noun('Knots Equivalent Airspeed', morph(9), fp(measure), fpcat(keas)).
noun('LAIM', morph(4), fp(missile), fpcat('AIM-9L')).
noun('Laser Guided Bomb', morph(9), fp(bomb), fpcat('LGB')).
noun('LCDR', morph(4), fp(rank)).
noun('LGB', morph(4), fp(bomb)).
noun('Linda', morph(9), fp('human being')).
noun('LLLGB', morph(4), fp(bomb)).
noun('latch assembly', morph(3), fp(phys_obj)).
noun('load test', morph(1), fp(event)).
noun('Lockheed', morph(9), fp(organization)).
noun('Long Wavelength Infrared', morph(9), fp(instrument), fpcat('LWIR detector')).
noun('Low Level Laser Guided Bomb', morph(9), fp(bomb), fpcat('LLLGB')).
noun('Lt', morph(4), fp(rank), fpcat('Lieutenant')).
noun('LWIR', morph(4), fp(instrument), fpcat('LWIR detector')).
noun('Machine Shop', morph(9), fp(location)).
noun('MAD', morph(4), fp(organization)).
noun('Major', morph(9), fp(rank)).
noun('Marine Air Detachment', morph(9), fp(organization), fpcat('MAD')).
noun('Marines', morph(9), fp(organization)).
noun('Mark 7', morph(4), fp(vehicle), fpcat('Mark-7')).
noun('Mark-7', morph(4), fp(vehicle)).
noun('Martin Marietta', morph(9), fp(organization)).
noun('Maverick', morph(9), fp(missile)).
noun('MAW/Marines', morph(9), fp(organization), fpcat('Marines')).
noun('Maxwell', morph(9), fp('human being')).
noun('MDS', morph(4), fp('#')).
noun('mercury vapor', morph(1), fp('chemical element')).
noun('Mich Lab', morph(4), fp(building), fpcat('Michelson Laboratory')).
noun('Mich', morph(4), fp('human being'), fpcat('Michelson')).
noun('Michelson Lab', morph(4), fp(building), fpcat('Michelson Laboratory')).
noun('Michelson', morph(9), fp('human being')).
noun('MIG-21', morph(4), fp(aircraft)).
noun('Missouri River', morph(9), fp(location)).
noun('Misty 13', morph(9), fp(marking)).
noun('MK 4', morph(4), fp(gun), fpcat('MK-4')).
noun('MK 7', morph(4), fp(vehicle), fpcat('Mark-7')).
noun('MK 81', morph(4), fp(bomb), fpcat('MK-81')).
noun('MK 82', morph(4), fp(bomb), fpcat('MK-82')).
noun('MK 91', morph(4), fp(bomb), fpcat('MK-91')).
noun('MK-4', morph(4), fp(gun)).
noun('MK-7', morph(4), fp(vehicle), fpcat('Mark-7')).
noun('MK-81', morph(4), fp(bomb)).
noun('MK-82', morph(4), fp(bomb)).
noun('MK-91', morph(4), fp(bomb)).
noun('modified Sidewinder 9C', morph(9), fp(missile), fpcat('modified AIM-9C')).
noun('NA-3B', morph(4), fp(aircraft), fpcat('A-3B')).
noun('Naces', morph(9), fp(organization)).
noun('NAF', morph(4), fp(location), fpcat('Armitage Field')).
noun('NASA', morph(4), fp(organization)).
noun('Naval Weapons Center', morph(9), fp(organization), fpcat('NWC')).
noun('NG', morph(4), fp(id)).
noun('night attack', morph(1), fp(event)).
noun('night hawk', morph(9), fp(marking)).
noun('night vision', morph(7), fp(abst_obj)).
noun('Northrop', morph(9), fp(organization)).
noun('NRA-3B', morph(4), fp(aircraft), fpcat('A-3B')).
noun('NWC', morph(4), fp(organization)).
noun('Olancha Peak', morph(9), fp(location)).
```

```
noun('OPNAV', morph(4), fp(organization)).
noun('OV-102', morph(4), fp(ship)).
noun('overall view', morph(1), fp(event)).
noun('overhead view', morph(1), fp(event)).
noun('Pacific', morph(9), fp(ocean), fpcat('Pacific Ocean')).
noun('Panamint Valley', morph(9), fp(location)).
noun('Pax River', morph(9), fp(location)).
noun('PCH-1', morph(4), fp(ship)).
noun('Phoenix', morph(9), fp(missile)).
noun('photovoltaic cell', morph(1), fp(generator)).
noun('PMTC', morph(4), fp(organization)).
noun('Point Mugu', morph(9), fp(location)).
noun('Puget Sound', morph(9), fp(locaton)).
noun('pusher sled', morph(1), fp(vehicle)).
noun('QF-4', morph(4), fp(aircraft)).
noun('QF-4B', morph(4), fp(aircraft)).
noun('QF-4H-1', morph(4), fp(aircraft)).
noun('QF-9', morph(4), fp(aircraft)).
noun('QF-9F', morph(4), fp(aircraft)).
noun('QF-86', morph(4), fp(aircraft)).
noun('QF4H-1', morph(4), fp(aircraft), fpcat('QF-4H-1')).
noun('R. W. Strongback', morph(9), fp('human being')).
noun('RA-5C', morph(4), fp(aircraft)).
noun('Range Control Center', morph(9), fp(phys_obj), fpcat('RCC')).
noun('RAPEC', morph(4), fp(project)).
noun('RCC', morph(4), fp(phys_obj)).
noun('RD', morph(4), fp('#')).
noun('rear view', morph(1), fp(event)).
noun('Richard', morph(9), fp('human being')).
noun('Ridgecrest', morph(9), fp(location)).
noun('Robert', morph(9), fp('human being')).
noun('rocket burn', morph(7), fp(event)).
noun('rocket engine', morph(7), fp(phys_obj)).
noun('rocket motor', morph(7), fp(phys_obj), fpcat('rocket engine')).
noun('Rockeye II', morph(9), fp(bomb)).
noun('Rockwell International', morph(9), fp(organization)).
noun('Roseville', morph(9), fp(location)).
noun('rotary wing', morph(1), fp(phys_obj)).
noun('rotary wing dispenser', morph(1), fp(phys_obj)).
noun('RTV', morph(4), fp('#')).
noun('Rzeszotko', morph(9), fp('human being')).
noun('Samson', morph(9), fp(drone)).
noun('S-2', morph(4), fp(aircraft)).
noun('S-2A', morph(4), fp(aircraft)).
noun('SAR', morph(4), fp(instrument)).
noun('SCO-96', morph(4), fp(test)).
noun('SCO-146', morph(4), fp(test)).
noun('SCR 584', morph(4), fp('#'), fpcat('SCR-584')).
noun('SCR-584', morph(4), fp('#')).
noun('sea launch', morph(2), fp(event)).
noun('search set pod', morph(1), fp(pod)).
noun('Searles Lake', morph(9), fp(location)).
noun('self-solderability', morph(7), fp(event), fpcat(solder)).
noun('separation test', morph(1), fp(event)).
noun('Seventh Fleet', morph(9), fp(organization)).
noun('SH', morph(4), fp(id)).
noun('Shrike', morph(9), fp(missile)).
noun('side view', morph(1), fp(event)).
noun('Sidearm', morph(9), fp(missile)).
noun('Sidewinder 1A', morph(4), fp(missile), fpcat('AIM-9B')).
noun('Sidewinder 1C', morph(4), fp(missile), fpcat('AIM-9D')).
```

```
noun('Sidewinder 1C IRAH', morph(4), fp(missile), fpcat('AIM-9D')).
noun('Sidewinder 1C SAR', morph(4), fp(missile), fpcat('AIM-9C')).
noun('Sidewinder 9C', morph(4), fp(missile), fpcat('AIM-9C')).
noun('Sidewinder 9M', morph(4), fp(missile), fpcat('AIM-9M')).
noun('Sidewinder', morph(9), fp(missile)).
noun('Sierra Nevada', morph(9), fp(location)).
noun('Sierra', morph(9), fp(location), fpcat('Sierra Nevada')).
noun('SIIIS-ER', morph(4), fp('#')).
noun('Silver Bullet', morph(9), fp(bomb)).
noun('Sinkex', morph(9), fp(operation)).
noun('Skipper', morph(9), fp(bomb)).
noun('Skyhawk', morph(9), fp(aircraft)).
noun('Skyhawk II', morph(9), fp(aircraft), fpcat('Skyhawk 2')).
noun('Skyray', morph(9), fp(pod)).
noun('Skywarrior', morph(9), fp(aircraft)).
noun('sled test', morph(1), fp(event)).
noun('SN', morph(4), fp(id), fpcat(serial_no)).
noun('Snipe', morph(9), fp(missile)).
noun('Snort', morph(9), fp(location)).
noun('SOG', morph(4), fp('#')).
noun('solder board', morph(1), fp(phys_obj)).
noun('Sparrow', morph(9), fp(missile)).
noun('Sparrow 3', morph(9), fp(missile)).
noun('Sparrow III', morph(9), fp(missile), fpcat('Sparrow 3')).
noun('spin canister', morph(1), fp(phys_obj)).
noun('SRT-6', morph(4), fp(system)).
noun('SSG-N1', morph(4), fp(ship)).
noun('Standard Arm', morph(9), fp(missile)).
noun('Standard Missile II-N', morph(9), fp(missile), fpcat('Standard Arm II-N')).
noun('Standard Missile', morph(9), fp(missile), fpcat('Standard Arm')).
noun('static firing', morph(1), fp(event)).
noun('Stephen', morph(9), fp('human being')).
noun('straight on view', morph(1), fp(event), fpcat('front view')).
noun('STS-1', morph(4), fp('#')).
noun('Super Cobra', morph(9), fp(helicopter)).
noun('T-5', morph(4), fp(tower)).
noun('T-45', morph(4), fp(aircraft)).
noun('T-45A', morph(4), fp(aircraft)).
noun('Tacit/Rainbow', morph(9), fp(missile)).
noun('tail fin', morph(1), fp(phys_obj)).
noun('Target Detecting Device', morph(9), fp(equipment), fpcat('TDD')).
noun('TA-7C', morph(4), fp(aircraft), fpcat('A-7C')).
noun('TAV-8B', morph(4), fp(aircraft), fpcat('AV-8B')).
noun('TDD', morph(4), fp(equipment)).
noun('test range', morph(1), fp(range)).
noun('TH-1', morph(4), fp(aircraft)).
noun('TH-1L', morph(4), fp(aircraft)).
noun('TIM', morph(4), fp(equipment)).
noun('Tomahawk', morph(9), fp(missile)).
noun('Tomcat', morph(9), fp(aircraft)).
noun('total view', morph(1), fp(event)).
noun('TP', morph(4), fp(abst_obj), fpcat('test plan')).
noun('Tracking Instrumentation Mount', morph(9), fp(equipment), fpcat('TIM')).
noun('tracking mount', morph(1), fp(equipment)).
noun('Trident', morph(9), fp(missile)).
noun('UH-1', morph(4), fp(aircraft)).
noun('UH-1E', morph(4), fp(aircraft)).
noun('UH-1N', morph(4), fp(aircraft)).
noun('UH-2', morph(4), fp(aircraft)).
noun('UH-2A', morph(4), fp(aircraft)).
noun('USAF#', morph(4), fp(id), fpcat('USAF_no')).
```

```
noun('USAF', morph(4), fp(organization)).
noun('USMC', morph(4), fp(organization)).
noun('USN', morph(4), fp(organization)).
noun('USS Columbia', morph(9), fp(ship)).
noun('USS Forrestal', morph(9), fp(ship)).
noun('USS High Point', morph(9), fp(ship)).
noun('USS Kitty Hawk', morph(9), fp(ship)).
noun('USS Lexington', morph(9), fp(ship)).
noun('USS Repertus', morph(9), fp(ship)).
noun('V/STOL', morph(9), fp(aircraft), fpcat('V/STOL aircraft')).
noun('VC-3', morph(4), fp(organization)).
noun('video screen', morph(1), fp(phys_obj)).
noun('video view', morph(1), fp(event)).
noun('VLA', morph(4), fp(engine)).
noun('VMA-513', morph(4), fp(organization)).
noun('VX-5', morph(4), fp(organization)).
noun('Walleye II', morph(9), fp(bomb)).
noun('Walleye', morph(9), fp(bomb)).
noun('Weapons Survivability', morph(9), fp(laboratory), fpcat('Weapons Survivability
Laboratory')).
noun('Weston', morph(9), fp('human being')).
noun('WF', morph(4), fp(id)).
noun('Wild Weasel', morph(9), fp(aircraft)).
noun('Wincn', morph(9), fp('human being')).
noun('WSSA', morph(4), fp(phys_obj)).
noun('XAIM 54', morph(4), fp(missile), fpcat('AIM-54')).
noun('XAIM 54C', morph(4), fp(missile), fpcat('AIM-54C')).
noun('XAIM-54', morph(4), fp(missile)).
noun('XAIM-54C', morph(4), fp(missile)).
noun('XE', morph(4), fp(id)).
noun('XN-01', morph(4), fp(phys_obj)).
noun('YA-4C', morph(4), fp(aircraft), fpcat('A-4C')).
noun('Young', morph(9), fp('human being')).
noun(action, morph(1), fp(event), fpcat(operation)).
noun(air, morph(6), fp(location)).
noun(aircraft, morph(1), fp(phys_obj)).
noun(airfield, morph(1), fp(location), fpcat(airport)).
noun(airport, morph(1), fp(location)).
noun(angle, morph(1), fp(abst_obj)).
noun(antenna, morph(1), fp(phys_obj)).
noun(area, morph(1), fp(location), fpcat(location)).
noun(art, morph(1), fp(abst_obj)).
noun(artist, morph(1), fp('human being')).
noun(attack, morph(1), fp(event)).
noun(attempt, morph(1), fp(event)).
noun(assembly, morph(3), fp(event)).
noun(aviator, morph(1), fp('human being')).
noun(background, morph(7), fp(position)).
noun(bank, morph(1), fp(abst_obj)).
noun(battery, morph(3), fp(phys_obj), fpcat('electric battery')).
noun(bay, morph(1), fp(location)).
noun(blast, morph(1), fp(event), fpcat(explosion)).
noun(blaze, morph(1), fp(mark)).
noun(bldg, morph(4), fp(phys_obj), fpcat(building)).
noun(bloom, morph(1), fp(event)).
noun(board, morph(1), fp(phys_obj)).
noun(body, morph(3), fp(phys_obj)).
noun(bomb, morph(1), fp(phys_obj)).
noun(booster, morph(1), fp(phys_obj), fpcat(explosive_booster)).
noun(bottom, morph(1), fp(position)).
noun(boxcar, morph(1), fp(vehicle)).
```

```
noun(boy, morph(1), fp('human being')).
noun(building, morph(1), fp(phys_obj)).
noun(bulkhead, morph(1), fp(phys_obj)).
noun(burn, morph(1), fp(event)).
noun(butt, morph(1), fp(position), fpcat(end)).
noun(camera, morph(1), fp(equipment)).
noun(can, morph(1), fp(phys_obj)).
noun(canister, morph(1), fp(phys_obj), fpcat(can)).
noun(canopy, morph(3), fp(phys_obj)).
noun(center, morph(1), fp(position)).
noun(charge, morph(1), fp(weapon), fpcat(weapon)).
noun(checkerboard, morph(1), fp(art)).
noun(chemical, morph(1), fp(phys_obj), fpcat('chemical element')).
noun(chip, morph(1), fp(phys_obj)).
noun(chute, morph(1), fp(phys_obj), fpcat(parachute)).
noun(clock, morph(1), fp(phys_obj)).
noun(cloud, morph(1), fp(phys_obj)).
noun(cockpit, morph(1), fp(phys_obj)).
noun(comparison, morph(1), fp(event)).
noun(composite, morph(1), fp(phys_obj)).
noun(compound, morph(1), fp(phys_obj)).
noun(computer, morph(1), fp(equipment)).
noun(conception, morph(1), fp(event)).
noun(configuration, morph(1), fp(event)).
noun(connector, morph(1), fp(phys_obj)).
noun(console, morph(1), fp(equipment)).
noun(construction, morph(7), fp(event)).
noun(control, morph(1), fp(phys_obj), fpcat('control equipment')).
noun(cookoff, morph(1), fp(event), fpcat('cookoff test')).
noun(crane, morph(1), fp(phys_obj)).
noun(crew, morph(8), fp('human being'), fpcat('human being')).
noun(crewmen, morph(8), fp('human being'), fpcat('human being')).
noun(damage, morph(1), fp(event)).
noun(debris, morph(6), fp(phys_obj)).
noun(decoy, morph(1), fp(phys_obj)).
noun(degree, morph(1), fp(measure)).
noun(deployment, morph(1), fp(event), fpcat(spread)).
noun(desert, morph(1), fp(location)).
noun(dispenser, morph(1), fp(phys_obj)).
noun(dolly, morph(3), fp(vehicle)).
noun(dome, morph(1), fp(phys_obj)).
noun(drogue, morph(1), fp(phys_obj), fpcat('drogue parachute')).
noun(drone, morph(1), fp(phys_obj)).
noun(drop, morph(1), fp(event), fpcat(launch)).
noun(dummy, morph(3), fp(phys_obj)).
noun(duration, morph(1), fp(measure)).
noun(edge, morph(1), fp(position)).
noun(education, morph(1), fp(event)).
noun(end, morph(1), fp(position)).
noun(ejection, morph(1), fp(event)).
noun(elevation, morph(1), fp(location)).
noun(elevator, morph(1), fp(phys_obj)).
noun(employee, morph(1), fp('human being'), fpcat('human being')).
noun(equipment, morph(1), fp(phys_obj)).
noun(erosion, morph(7), fp(event)).
noun(evaluation, morph(1), fp(event)).
noun(evening, morph(1), fp(time)).
noun(event, morph(1), fp(abst_obj)).
noun(examination, morph(1), fp(event), fpcat(inspection)).
noun(example, morph(1), fp(abst_obj)).
noun(exhaust, morph(1), fp(phys_obj)).
```

```
noun(explosion, morph(1), fp(event)).
noun(facility, morph(3), fp(phys_obj), fpcat(equipment)).
noun(farm, morph(1), fp(location)).
noun(feet, morph(6), fp(measure)).
noun(field, morph(1), fp(location), fpcat(airport)).
noun(fin, morph(1), fp(phys_obj)).
noun(fire, morph(1), fp(phys_obj)).
noun(fireball, morph(1), fp(phys_obj)).
noun(firing, morph(1), fp(event), fpcat(launch)).
noun(flight, morph(1), fp(event)).
noun(flightline, morph(1), fp(abst_obj), fpcat('flight path')).
noun(flower, morph(1), fp(plant)).
noun(flyover, morph(1), fp(event), fpcat(flight)).
noun(foreground, morph(7), fp(position)).
noun(formation, morph(1), fp(abst_obj)).
noun(frame, morph(1), fp(phys_obj)).
noun(front, morph(1), fp(position)).
noun(fuel, morph(1), fp(phys_obj)).
noun(girl, morph(1), fp('human being')).
noun(goggles, morph(8), fp(phys_obj)).
noun(ground, morph(1), fp(location), fpcat(land)).
noun(gun, morph(1), fp(weapon)).
noun(hangar, morph(1), fp(building)).
noun(haze, morph(1), fp(phys_obj)).
noun(heat, morph(6), fp(abst_obj)).
noun(helicopter, morph(1), fp(aircraft)).
noun(heliostat, morph(1), fp(instrument)).
noun(helo, morph(1), fp(aircraft), fpcat(helicopter)).
noun(hoisting, morph(1), fp(event)).
noun(horse, morph(1), fp(animal)).
noun(hydrofoil, morph(1), fp(ship), fpcat('hydrofoil craft')).
noun(ignition, morph(1), fp(event)).
noun(impact, morph(1), fp(event), fpcat(hit)).
noun(incident, morph(1), fp(abst_obj), fpcat(event)).
noun(ingestion, morph(7), fp(event)).
noun(inspection, morph(7), fp(event)).
noun(instrumentation, morph(6), fp(phys_obj), fpcat(instrument)).
noun(intensity, morph(3), fp(abst_obj)).
noun(intercept, morph(1), fp(event), fpcat(hit)).
noun(inverter, morph(1), fp(equipment)).
noun(keas, morph(4), fp(measure)).
noun(kind, morph(1), fp(abst_obj)).
noun(knozy, morph(7), fp(equipment), fpcat('homing device')).
noun(kts, morph(4), fp(measure)).
noun(lab, morph(1), fp(building), fpcat(laboratory)).
noun(laboratory, morph(1), fp(building)).
noun(land, morph(1), fp(location)).
noun(laser, morph(1), fp(phys_obj)).
noun(latch, morph(2), fp(phys_obj), fpcat(fastener)).
noun(launch, morph(1), fp(event)).
noun(launcher, morph(1), fp(phys_obj)).
noun(light, morph(1), fp(phys_obj), fpcat(luminaire)).
noun(lighting, morph(6), fp(phys_obj), fpcat(luminaire)).
noun(line, morph(1), fp(marking)).
noun(load, morph(1), fp(phys_obj), fpcat(payload)).
noun(loader, morph(1), fp(vehicle)).
noun(logo, morph(1), fp(abst_obj), fpcat(marking)).
noun(marking, morph(1), fp(abst_obj)).
noun(midst, morph(1), fp(position), fpcat(center)).
noun(mission, morph(1), fp(abst_obj)).
noun(missile, morph(1), fp(phys_obj)).
```

```
noun(mm, morph(4), fp(measure)).
noun(model, morph(1), fp(phys_obj)).
noun(motor, morph(1), fp(phys_obj), fpcat(engine)).
noun(mountain, morph(1), fp(location)).
noun(night, morph(1), fp(time), fpcat(evening)).
noun(nose, morph(1), fp(phys_obj)).
noun(number, morph(1), fp(abst_obj)).
noun(opening, morph(1), fp(abst_obj)).
noun(operation, morph(1), fp(event)).
noun(operator, morph(1), fp(phys_obj)).
noun(overview, morph(1), fp(event)).
noun(orbitor, morph(1), fp(vehicle)).
noun(ordnance, morph(1), fp(phys_obj)).
noun(paint, morph(1), fp(phys_obj)).
noun(pallet, morph(1), fp(phys_obj)).
noun(panel, morph(1), fp(phys_obj)).
noun(parachute, morph(1), fp(phys_obj)).
noun(part, morph(1), fp('#')).
noun(pattern, morph(1), fp(abst_obj)).
noun(people, morph(8), fp('human being'), fpcat('human being')).
noun(peripheral, morph(1), fp(phys_obj), fpcat(equipment)).
noun(personnel, morph(8), fp('human being'), fpcat('human being')).
noun(pedestal, morph(1), fp(phys_obj)).
noun(photograph, morph(1), fp(event), fpcat(view)).
noun(photovoltaic, morph(1), fp(generator), fpcat('photovoltaic cell')).
noun(picture, morph(1), fp(event), fpcat(view)).
noun(pilot, morph(1), fp(phys_obj)).
noun(plaque, morph(1), fp(phys_obj)).
noun(plane, morph(1), fp(phys_obj), fpcat(aircraft)).
noun(plume, morph(1), fp(phys_obj)).
noun(pod, morph(1), fp(phys_obj)).
noun(pole, morph(1), fp(phys_obj)).
noun(port, morph(1), fp(abst_obj), fpcat(opening)).
noun(position, morph(1), fp(abst_obj)).
noun(power, morph(1), fp(abst_obj)).
noun(practice, morph(1), fp(abst_obj)).
noun(practise, morph(1), fp(abst_obj)).
noun(production, morph(1), fp(event)).
noun(program, morph(1), fp(abst_obj)).
noun(progress, morph(7), fp(abst_obj)).
noun(project, morph(1), fp(abst_obj)).
noun(projectile, morph(1), fp(phys_obj)).
noun(pylon, morph(1), fp(phys_obj)).
noun(rack, morph(1), fp(phys_obj)).
noun(range, morph(1), fp(location)).
noun(radar, morph(1), fp(phys_obj)).
noun(radome, morph(1), fp(phys_obj)).
noun(rd, morph(4), fp(event), fpcat(round)).
noun(rear, morph(6), fp(position)).
noun(red, morph(6), fp(color)).
noun(remains, morph(8), fp(phys_obj)).
noun(resistor, morph(1), fp(phys_obj)).
noun(revetment, morph(1), fp(phys_obj)).
noun(river, morph(1), fp(location)).
noun(rock, morph(1), fp(phys_obj)).
noun(rocket, morph(1), fp(phys_obj)).
noun(room, morph(1), fp(phys_obj)).
noun(round, morph(1), fp(event)).
noun(run, morph(1), fp(event)).
noun(runway, morph(1), fp(location)).
noun(sea, morph(1), fp(location)).
```

```
noun(seat, morph(1), fp(phys_obj)).
noun(seeker, morph(1), fp(equipment), fpcat('homing device')).
noun(sequence, morph(1), fp(abst_obj)).
noun(setup, morph(1), fp(abst_obj)).
noun(ship, morph(1), fp(phys_obj)).
noun(shot, morph(1), fp(event)).
noun(shuttle, morph(1), fp(vehicle)).
noun(side, morph(1), fp(position)).
noun(silver, morph(7), fp(color)).
noun(simulator, morph(1), fp(relation)).
noun(site, morph(1), fp(location), fpcat(location)).
noun(skid, morph(1), fp(phys_obj)).
noun(sled, morph(1), fp(vehicle)).
noun(smoke, morph(6), fp(phys_obj)).
noun(snow, morph(6), fp(phys_obj)).
noun(solder, morph(1), fp(phys_obj)).
noun(soldering, morph(7), fp(event)).
noun(sort, morph(1), fp(abst_obj)).
noun(spreading, morph(7), fp(event)).
noun(stand, morph(1), fp(phys_obj)).
noun(station, morph(1), fp(phys_obj)).
noun(strobe, morph(1), fp(equipment)).
noun(structure, morph(1), fp(phys_obj), fpcat(phys_obj)).
noun(study, morph(3), fp(abst_obj)).
noun(submarine, morph(1), fp(ship)).
noun(submissile, morph(1), fp(missile), fpcat(missile)).
noun(sunset, morph(1), fp(time)).
noun(survivability, morph(6), fp(event)).
noun(system, morph(1), fp(phys_obj)).
noun(tail, morph(1), fp(phys_obj)).
noun(takeoff, morph(1), fp(event)).
noun(tank, morph(1), fp(phys_obj)).
noun(target, morph(1), fp(phys_obj)).
noun(terrain, morph(1), fp(location), fpcat(land)).
noun(test, morph(1), fp(event)).
noun(testing, morph(1), fp(event), fpcat(test)).
noun(tilt, morph(1), fp(abst_obj)).
noun(top, morph(1), fp(position)).
noun(tower, morph(1), fp(phys_obj)).
noun(tp, morph(1), fp(abst_obj), fpcat('test plan')).
noun(track, morph(1), fp(phys_obj)).
noun(trail, morph(1), fp(phys_obj)).
noun(trailer, morph(1), fp(vehicle)).
noun(training, morph(6), fp(event), fpcat(education)).
noun(tree, morph(1), fp(plant)).
noun(truck, morph(1), fp(vehicle)).
noun(type, morph(1), fp(abst_obj), fpcat(kind)).
noun(underside, morph(7), fp(position)).
noun(validation, morph(1), fp(event)).
noun(valley, morph(3), fp(location)).
noun(van, morph(1), fp(vehicle), fpcat(truck)).
noun(view, morph(1), fp(event)).
noun(warhead, morph(1), fp(phys_obj)).
noun(wash, morph(2), fp(location)).
noun(water, morph(1), fp(phys_obj)).
noun(weapon, morph(1), fp(phys_obj)).
noun(wildflower, morph(1), fp(plant)).
noun(wing, morph(1), fp(phys_obj)).
noun(wingtip, morph(1), fp(phys_obj)).
noun(wire, morph(1), fp(phys_obj)).
noun(worker, morph(1), fp('human being'), fpcat('human being')).
```

```
numadj('#E').
numadj('#11').

numadv('at least').
numadv(about).
numadv(approximately).
numadv(nearly).
numadv(only).

of(of).

order('front to back', fpcat(front_to_back)).
order('inside out', fpcat(inside_out)).
order('left to right', fpcat(left_to_right)).
order('right to left', fpcat(right_to_left)).
order('top down', fpcat(top_down)).
order('top to bottom', fpcat(top_down)).

ordinal('2nd', fp(order), fpcat(2)).
ordinal(eigth, fp(order), fpcat(8)).
ordinal(fifth, fp(order), fpcat(5)).
ordinal(first, fp(order), fpcat(1)).
ordinal(fourth, fp(order), fpcat(4)).
ordinal(nineth, fp(order), fpcat(9)).
ordinal(second, fp(order), fpcat(2)).
ordinal(seventh, fp(order), fpcat(7)).
ordinal(sixth, fp(order), fpcat(6)).
ordinal(tenth, fp(order), fpcat(10)).
ordinal(third, fp(order), fpcat(3)).

lparen('(').
rparen(')').

prep('away from', fpcat(from)).
prep('clear of', fp(event), fpcat(free)).
prep('except for').
%prep('from underside of', fpcat(under)).
prep('in front of', fpcat(before)).
prep('in on', fpcat(to)).
prep('in the middle of').
prep('inside of', fpcat(in)).
prep('next to', fpcat(next_to)).
prep('on top of', fp(rev), fpcat(under)).
%prep('on underside of', fpcat(under)).
prep('out of').
prep('outside of', fpcat(outside)).
prep('prior to', fp(time), fpcat(before)).
prep(aboard, fpcat(on)).
prep(about).
prep(above, fp(rev), fpcat(under)).
prep(across).
prep(after, fp(time)).
prep(against).
prep(along).
prep(alongside, fpcat(next_to)).
prep(around).
prep(at).
prep(before, fp(position)).
prep(behind).
prep(below, fpcat(under)).
```

```
prep(beneath, fpcat(under)).
prep(between).
prep(by).
prep(during).
prep(except).
%prep(facing).
prep(for).
prep(from).
prep(in).
prep(into, fpcat(on)).
prep(minus).
prep(near).
prep(off).
prep(on).
prep(onto, fpcat(on)).
prep(out).
prep(over).
prep(per).
prep(since).
prep(than).
prep(through).
prep(thru, fpcat(through)).
prep(til).
prep(to).
prep(toward, fpcat(to)).
prep(towards, fpcat(to)).
prep(under).
prep(underneath, fpcat(under)).
prep(until).
prep(up).
prep(upon).
prep(vicinity).
prep(with).
prep(within).
prep(without).

prep_noun(in, discontinuous([way])).
prep_noun(on, discontinuous([order])).
prep_noun(on, discontinuous([way])).

pronoun(he).
pronoun(her).
pronoun(him).
pronoun(i).
pronoun(it, fp(abst_obj)).
pronoun(its, fp(abst_obj)).
pronoun(me).
pronoun(she).
pronoun(them).
pronoun(they).
pronoun(us).
pronoun(we).
pronoun(ya).
pronoun(you).

punct('-').
punct(';').
punct(':').

quant('1/4').
quant('3/4').
```

```
quant('a couple').
quant('a few').
quant('a lot').
quant(all).
quant(another).
quant(any).
quant(both).
quant(closeup).
quant(complete, fpcat(full)).
quant(couple).
quant(each).
quant(either).
quant(entire, fpcat(all)).
quant(few).
quant(full).
quant(many, fpcat(plural)).
quant(multiple, fpcat(plural)).
quant(neither, fp(neg)).
quant(no, fp(neg)).
quant(none, fp(neg)).
quant(other).
quant(overall).
quant(several, fpcat(plural)).
quant(some, fpcat(plural)).

quantadv(anytime).
quantadv(anywhere).
quantadv(everywhere).
quantadv(nowhere, fp(neg)).
quantadv(sometime).
quantadv(somewhere).

quantpro('no one', fp(neg)).
quantpro(anybody).
quantpro(anyone).
quantpro(anything).
quantpro(everybody).
quantpro(everyone).
quantpro(everything).
quantpro(nobody, fp(neg)).
quantpro(nothing, fp(neg)).
quantpro(one).
quantpro(ones).
quantpro(somebody).
quantpro(someone).
quantpro(something).

quote('''').

reflxpro('each other').
reflxpro('one another').
reflxpro(herself).
reflxpro(himself).
reflxpro(itself).
reflxpro(myself).
reflxpro(ourself).
reflxpro(ourselves).
reflxpro(themselves).
reflxpro(yourself).
reflxpro(yourselves).
```

```
relpro(that).
relpro(where).
relpro(which).
relpro(who).

resp('10-4', fpcat(positive)).
resp('don''t know', fpcat(unknown)).
resp('i don''t know', fpcat(unknown)).
resp('thank you', fpcat(general)).
resp('that is a roger', fpcat(positive)).
resp('that is affirmative', fpcat(positive)).
resp('that is negative', fpcat(positive)).
resp('that''s a negative', fpcat(negative)).
resp('that''s a roger', fpcat(positive)).
resp('that''s a wilco', fpcat(positive)).
resp('that''s affirmative', fpcat(positive)).
resp('well done', fpcat(general)).
resp('will do', fpcat(positive)).
resp(affirmative, fpcat(positive)).
resp(allright, fpcat(positive)).
resp(certainly, fpcat(positive)).
resp(correct, fpcat(positive)).
resp(good, fpcat(general)).
resp(great, fpcat(general)).
resp(nah, fpcat(negative)).
resp(negative, fpcat(negative)).
resp(no, fpcat(negative)).
resp(okay, fpcat(positive)).
resp(roger, fpcat(positive)).
resp(sorry, fpcat(general)).
resp(wilco, fpcat(positive)).
resp(yeah, fpcat(positive)).
resp(yes, fpcat(positive)).

subconj('as soon as').
subconj('in order that').
subconj('in order').
subconj('so that').
subconj(after, fp(time)).
subconj(as).
subconj(because).
subconj(before, fp(position)).
subconj(during, fp(time)).
subconj(cause).
subconj(however).
subconj(if).
subconj(like).
subconj(once).
subconj(since).
subconj(so).
subconj(though).
subconj(unless).
subconj(until).
subconj(whether).
subconj(while).

timadv('right now').
timadv(now).
timadv(then).
timadv(today).
timadv(tomorrow).
```

```
timadv(tonight).
timadv(yesterday).

to('in order to').
to(to).
```

APPENDIX D

SYNTACTIC GRAMMAR

The majority of this grammar is based on the Original LSI DBG system [Montgomery 89].

```
%%% sccs('@(#)casegram.pl   1.19 12/30/91').
%%%
%%% Syntactic Grammar


%%% ---------------  Sentence  ---------------

s:1.00 ==> formula.
s:1.00 ==> formula, callsign.             %rsb
s:1.00 ==> callsign, formula.             %rsb
s:0.75 ==> callsign, question.            %rsb-low weight to avoid confusion with
declaratives
s:0.82 ==> question, callsign.            %rsb
s:0.93 ==> question.
s:0.90 ==> imperative.
s:0.82 ==> callsign, imperative.   %rsb - e.g. 2-100:08:16:34
s:0.80 ==> imperative, callsign.   %rsb
s:0.84 ==> declarative.
s:0.83 ==> callsign, declarative.  %rsb
s:0.75 ==> declarative, callsign.  %rsb
s:0.87 ==> phrase.
s:0.78 ==> callsign, phrase.              %rsb
s:0.95 ==> conj, question.         %rsb
s:0.90 ==> conj, imperative.              %rsb
s:0.87 ==> conj, declarative.             %rsb
s:0.70 ==> subconj, declarative.   %rsb - e.g. "Because I need him here"

%%% ---------------  Formula  ---------------

formula ==> com.                   %rsb - conv to com
formula ==> intro.
formula ==> resp.
formula ==> resp, phrase.
formula ==> exp.

%%% ---------------  Question  ---------------

question ==> qvp.
question ==> interpro, ofp, qvp.
question ==> interpro, ofp, vp.
question ==> interpro, ofp.
question ==> interpro, qvp.
question ==> interpro, vp.
question ==> interpro.

%%% ---------------  Imperative Sentence  ---------------
```

```
imperative ==> am, ivp.
imperative:0.90 ==> civp, conj, civp.

%%% --------------- Conjoined Imperative Sentence ---------------

civp ==> ivp.

%%% --------------- Declarative Sentence ---------------

%   for 'empty actor', RES
declarative(declarative(np('*empty*'), Vp))^[Vp] ==>
    vp(case:[actor(empty)]).
declarative ==> am, np, am, vp.
declarative ==> am, np, vp.
declarative ==> np, vp, conj, vp.    %ejg - "chute is above seat but not open"
declarative ==> np, vp.
declarative ==> cs, conj, cs.
declarative ==> cs, cs, conj, cs.

%%% --------------- Conjoined Sentence ---------------

cs ==> np, vp.
cs ==> am, np, vp.
cs ==> am, np, am, vp.
cs ==> qvp.          %rsb - 2-100:08:36:08

%%% --------------- Phrase ---------------

phrase ==> np.
phrase ==> pp.
phrase ==> ajc.              %rsb - "moving out at this time", etc. (0-subjects)
phrase ==> np, ajc.  %rsb - "target neutralized", etc. (0-copula)
phrase ==> ordseq.   %ejg - "left to right: Sidewinder, Harm, and Maverick"

%%% --------------- Ordered Sequence ---------------

ordseq ==> order, colon, seq.

%%% --------------- Grid Number ---------------

gridnum ==> integer.

%%% --------------- Coordinate ---------------

coordinate ==> integer, quote, dir, by, integer, quote, dir.

%%% --------------- Noun Phrase ---------------

np:1.00 ==> pronoun.
np:1.00 ==> quantpro, rnp.          %rsb - e.g. 4-100:12:06:47,
np:0.99 ==> quantpro.
np:0.98 ==> detg, ng, rnp.
np:0.96 ==> detg, ng.
np:0.95 ==> n, letter.                     %rsb - "TRP echo"
np:0.94 ==> detg, ng, detg, ng.
np:0.90 ==> ng, rnp.
np:0.89 ==> ng.
np:0.88 ==> detg, ng, detg, ng, rnp.       %rsb - e.g. 2-100:08:35:46
np:0.83 ==> detpro, rnp.
np:0.82 ==> demadv.                 %rsb - alt to expro.
```

```
np:0.76 ==> cnp, conj, cnp.
np:0.74 ==> detg, cng, conj, cng.
np:0.71 ==> cnp, comma, cnp, comma, conj, cnp.    %ejg
np:0.70 ==> cnp, conj, cnp, conj, cnp.
np:0.60 ==> posnoun.                  %rsb - "one of his"
np:0.58 ==> detg, ng, rnp, ajc.            %rsb - for np's with both pp's and rc's
following.
np:0.56 ==> ng, rnp, afc.
np:0.40 ==> detpro.                   % npTest deleted

%%% --------------- Conjoined Noun Phrase  ---------------

cnp ==> detg, ng, rnp.
cnp ==> detg, ng.
cnp ==> ng, rnp.
cnp ==> ng.
cnp:0.85 ==> n.                       %rsb - "whiskey and delta"
cnp:0.80 ==> detpro, rnp.

%%% --------------- Sequence Noun Phrase  ---------------

seq ==> cnp, comma, conj, cnp.
seq ==> cnp, comma, seq.
seq ==> cnp, conj, cnp.
seq ==> cnp.

%%% --------------- Appositives  ---------------

app ==> lparen, np, rparen.  %ejg - "Sidearm missile (modified AIM 9C)"

%%% --------------- Noun Phrase Test  ---------------

%   npTest: this stuff can't be done at present
npTest :-
   goal(noun),
   !,
   fail.

%npTest ==> noun, !, { fail }.
%npTest ==> adj, !, { fail }.
%npTest ==> numgr, !, { fail }.
%npTest ==> lexCat, !, { fail }.
%npTest ==> lexCat, !, { fail }.
%npTest ==> lexCat, !, { fail }.
%npTest ==> { true }.

%%% --------------- Determiner Group  ---------------

detg:1.00 ==> numadv, det, refadjp, numgr.
detg:0.96 ==> numadv, det, refadjp, numgr, refadjp.
detg:0.92 ==> numadv, det, refadjp.
detg:0.88 ==> numadv, det, dirg.
detg:0.84 ==> numadv, det, numgr.
detg:0.80 ==> numadv, det, numgr, refadjp.
detg:0.75 ==> numadv, numgr.

detg:1.00 ==> numadv, refadjp, numgr.
detg:0.95 ==> numadv, refadjp, numgr, refadjp.
detg:0.90 ==> numadv, refadjp.
detg:0.85 ==> numadv, dirg.
detg:0.80 ==> numadv, numgr, refadjp.
```

```
detg:1.00 ==> det, refadjp, numgr.
detg:0.96 ==> det, refadjp, numgr, refadjp.
detg:0.92 ==> det, refadjp.
detg:0.88 ==> det, dirg.
detg:0.84 ==> det, numgr.
detg:0.80 ==> det, numgr, refadjp.

detg:1.00 ==> refadjp, numgr.
detg:0.95 ==> refadjp, numgr, refadjp.
detg:0.90 ==> refadjp.
detg:0.85 ==> dirg.
detg:0.85 ==> numgr.
detg:0.80 ==> numgr, refadjp.

detg:1.00 ==> det.
detg:0.95 ==> quant, det.    %rsb-e.g."all my people are stationary"

%%% --------------- Determiner Pronoun ---------------

detpro:1.00 ==> numadv, det, refadjp, numgr.
detpro:0.96 ==> numadv, det, refadjp, numgr, refadjp.
detpro:0.92 ==> numadv, det, refadjp.
detpro:0.88 ==> numadv, det, dirp.
detpro:0.84 ==> numadv, det, numgr.
detpro:0.80 ==> numadv, det, numgr, refadjp.

detpro:1.00 ==> numadv, refadjp, numgr.
detpro:0.96 ==> numadv, refadjp, numgr, refadjp.
detpro:0.92 ==> numadv, refadjp.
detpro:0.88 ==> numadv, dirp.
detpro:0.84 ==> numadv, numgr.
detpro:0.80 ==> numadv, numgr, refadjp.

detpro:1.00 ==> det, refadjp, numgr.
detpro:0.96 ==> det, refadjp, numgr, refadjp.
detpro:0.92 ==> det, refadjp.
detpro:0.88 ==> det, dirp.
detpro:0.84 ==> det, numgr.
detpro:0.80 ==> det, numgr, refadjp.

detpro:1.00 ==> refadjp, numgr.
detpro:0.97 ==> refadjp, numgr, refadjp.
detpro:0.94 ==> ordinal.
detpro:0.92 ==> refadj.
detpro:0.91 ==> dirp.
detpro:0.40 ==> numgr.
detpro:0.35 ==> numgr, refadjp.
detpro:0.82 ==> dempro.
detpro:0.80 ==> quant.

%%% --------------- Determiner ---------------

det ==> art.
det ==> dempro.
det ==> quant.
det ==> pospro.
det ==> pospro, own.
det ==> interpro.          %rsb - e.g. 4-100:12:20:55,32:00,39:52

%%% --------------- Number Group ---------------
```

```
% \+ Numgr2 = []  deleted from some; effectively becomes
% fract, tprep, num.
% fract, evaladv, num.
% fract.
% tprep, num.
% evaladv, num.
% can't be []

% numgrTest deleted from all below

numgr ==> num.

numgr:1.00 ==> numadv, num, fract, tprep, num.
numgr:0.95 ==> numadv, num, fract, evaladv, num.
numgr:0.90 ==> numadv, num, fract.
numgr:0.85 ==> numadv, num, tprep, num.
numgr:0.80 ==> numadv, num, evaladv, num.

numgr:1.00 ==> num, fract, tprep, num.
numgr:0.95 ==> num, fract, evaladv, num.
numgr:0.90 ==> num, fract.
numgr:0.85 ==> num, tprep, num.
numgr:0.80 ==> num, evaladv, num.

%%% ---------------  Number Group Test  ---------------

%numgrTest ==> noun, !, { fail }.
%numgrTest ==> noun, !, { fail }.
%numgrTest ==> { true }.

%%% ---------------  Number  ---------------

num ==> float.
num ==> integer.
num ==> cardinal.
num ==> numadj.                      %ejg - "target #E"

%%% ---------------  Integer  ---------------

integer ==> integer, correc, integer.          %rsb

%%% ---------------  Reference Adjective Phrase  ---------------

refadjp ==> ordinal.                             % refadjTest deleted
refadjp ==> refadj.
refadjp ==> evaladj.

%%% ---------------  Reference Adjective Test  ---------------

%refadjTest ==> noun, !, { fail }.
%refadjTest ==> { true }.

%%% ---------------  Noun Group  ---------------

ng:1.00 ==> adjp, ng.                %rsb - for common adjectival modifiers
ng:0.98 ==> n, num, ng.              %ejg - "bu# 163284 aircraft"
ng:0.97 ==> n, num, app.     %ejg - "test 1 (fast cookoff)"
ng:0.95 ==> n, num.          %rsb - "Grid 987987" (ejg)
ng:0.95 ==> n, letter, app.  %ejg - "strobe type s."
ng:0.94 ==> n, letter.               %ejg - "strobe type s."
```

```
ng:0.98 ==> n, app, ng.          %ejg - "Sidearm (modified Aim 9C) firing test"
ng:0.95 ==> n, app.              %ejg - "Sidearm (modified AIM 9C) on a stand"
ng:0.85 ==> n, ng.           %compounds
ng:0.84 ==> n.

%%% --------------- Conjoined Noun Group ---------------

cng:1.00 ==> n, ng.
cng:0.80 ==> n.

%%% --------------- Noun ---------------

n ==> noun.
n ==> letter.              %ejg - "strobe type s."
n ==> callsign, callnum.
n ==> coordinate.
n ==> date.
n:0.95 ==> callsign.       %rsb - 4-100:12:05:35
n:0.95 ==> n, gen, ng.            %rsb - "quebec's vehicle"
n:0.98 ==> dir.                   %rsb - 4-100:12:05:35,100:12:07:44
n:0.98 ==> loc.                   %ejg
n ==>  nounConj(F), conj, nounConj(F).
n:1.00 ==> ordinal, noun(ordnoun).
n:1.00 ==> numgr, noun(unit).
n:0.65 ==> adjp.
n ==> coord, coord, evaldj.

%%% --------------- Noun Conjunct ---------------

nounConj(F) ==> noun(F).

%%% --------------- Callsign ---------------

callsign ==> callsign, callsign.    %rsb - multiple cs's as vocatives
callsign:0.80 ==> integer.        %rsb - mostly for vocative, abbreviated cs's

%%% --------------- Letter ---------------

letter ==> letter, letter.        %rsb - "TRP echo echo"

%%% --------------- Adjective Phrase ---------------

adjp ==> adv, adjp.
adjp ==> diradj.                  %rsb - e.g. "the left side"
adjp ==> adj.
adjp ==> evaladj.
adjp ==> ordinal.
adjp ==> quant.                          %ejg - e.g., "3/4 front view"

%%% --------------- Relative Noun Phrase ---------------

rnp ==> ofp.
rnp ==> ajc, conj, ajc.
rnp ==> ajc.
rnp ==> pp.                       %ejg - commented out following line
rnp ==> am.                       %rsb - e.g. subj. np containing a pp
rnp ==> neg, ajc.
%rnp ==> rc.
rnp:0.60 ==> rc.
rnp:1.00 ==> dirp, ajc.
rnp:0.80 ==> dirp.
```

```
%%% --------------- Of Phrase ---------------

ofp ==> of, np.

%%% --------------- AJC ---------------

%   RES, 12/7/89, for "... moving out ..."
% ajc(ajc(Prpt, part(Adv), Rvp))^[Prpt, adv(Adv), Rvp] ==>
%    prpt(case:[vpart(P)]), adv(P), rvp.

ajc(ajc(V, Np-vcase(C), Rvp))^[V, Np, Rvp] ==>
    prpt(case:[dobj(phys_obj), vcase(C)]), np, rvp.
ajc(ajc(V, Np-vcase(C)))^[V, Np]:0.95 ==>
    prpt(case:[dobj(phys_obj), vcase(C)]), np.
ajc(ajc(V, Np-vcase(theme), Rvp))^[V, Np, Rvp]:0.95 ==>
    prpt(case:[dobj(phys_obj)]), np, rvp.
ajc(ajc(V, Np-vcase(theme)))^[V, Np] ==>
    prpt(case:[dobj(phys_obj)]), np.
ajc(ajc(Prpt, Prep, Rvp-vcase(C))-vtype(T))^[Prpt, Prep, Rvp] ==>
    prpt(case:[vprep(P),vtype(T),vcase(C)]), prep(P), rvp.

ajc(ajc(V, Dir-vcase(direction), Rvp))^[V, Dir, Rvp] ==>
    prpt(case:[dobj(direction)]), diradj, rvp.

ajc(ajc(V, Np-vcase(C), Rvp))^[V, Np, Rvp] ==>
    ppt(case:[dobj(phys_obj), vcase(C)]), np, rvp.
ajc(ajc(V, Np-vcase(C)))^[V, Np]:0.95 ==>
    ppt(case:[dobj(phys_obj), vcase(C)]), np.
ajc(ajc(V, Np-vcase(theme), Rvp))^[V, Np, Rvp] ==>
    ppt(case:[dobj(phys_obj)]), np, rvp.
ajc(ajc(V, Np-vcase(theme)))^[V, Np]:0.95 ==>
    ppt(case:[dobj(phys_obj)]), np.
ajc(ajc(V, Prep, Rvp-vcase(C))-vtype(T))^[V, Prep, Rvp] ==>
    ppt(case:[vprep(P),vtype(T),vcase(C)]), prep(P), rvp.

ajc:1.00 ==> timadv, prpt, rvp.
ajc:0.90 ==> timadv, ppt, rvp.
ajc:0.80 ==> timadv, having, ppt, rvp.
ajc ==> adv, ppt, rvp.
ajc:0.90 ==> adjp, am.
ajc ==> vnom, rvp.
ajc ==> vnom.                %ejg - e.g. "helicopter lifting off."
ajc:0.90 ==> prpt, by, rvp.
ajc:0.90 ==> prpt, rvp.
ajc:0.93 ==> ppt, rvp.
ajc:0.70 ==> prpt, by.               %rsb - e.g. "Moving."
ajc:0.70 ==> prpt.           %rsb - e.g. "Moving."
ajc:0.90 ==> ppt, ordseq.    %ejg - e.g. "weapons loaded top to bottom: ..."
ajc:0.87 ==> ppt.            %rsb - e.g. "Target neutralized."
ajc:0.86 ==> having, ppt, rvp.
ajc:0.80 ==> having, been, ppt, am.
ajc ==> evaladv, np.

%%% --------------- Verb Nominals ---------------

vnom(vnom(Prpt, part(Adv))-vtype(T))^[Prpt, adv(Adv)] ==>
    prpt(case:[vpart(P),vtype(T)]), adv(P).
vnom(vnom(Prpt, part(Adv)))^[Prpt, adv(Adv)] ==>
    prpt(case:[vpart(P)]), adv(P).
```

```
% Handle past participle also, not just present participle
% e.g., "helicopter lifted off"

vnom(vnom(Ppt, part(Adv))-vtype(T))^[Ppt, adv(Adv)] ==>
    ppt(case:[vpart(P),vtype(T)]), adv(P).
vnom(vnom(Ppt, part(Adv)))^[Ppt, adv(Adv)] ==>
    ppt(case:[vpart(P)]), adv(P).

%%% ---------------  Relative Clause  ---------------

rc ==> relpro, vp.
rc ==> relpro, np, am, vp.       %rsb - changed s to np-am-vp, etc.
rc ==> relpro, np, vp.
rc:0.80 ==> np, am, vp.                 %rsb - e.g. ?
rc:0.70 ==> np, vp.

%%% ---------------  Direction Phrase  ---------------

dirp ==> numadv, dirg, ofp.
dirp ==> dirg, ofp.

%%% ---------------  Direction Group  ---------------

dirg ==> dir.
dirg ==> dir, conj, dir.

%%% ---------------  Verb Phrase  ---------------

vp(vp(V, Np-vcase(C), Rvp))^[V, Np, Rvp] ==>
    v(case:[dobj(phys_obj), vcase(C)]), np, rvp.
vp(vp(V, Np-vcase(theme), Rvp))^[V, Np, Rvp] ==>
    v(case:[dobj(phys_obj)]), np, rvp.
vp(vp(V, Np-D, Rvp))^[V, Np, Rvp] ==>
    v(case:[dobj(D)]), { D \== phys_obj }, np, rvp.        % RES
%   For "you have got a bmp moving out to your south"
vp(vp(V, Np-vcase(C)))^[V, Np] ==>
    v(case:[dobj(phys_obj), vcase(C)]), np.
vp(vp(V, Np-vcase(theme)))^[V, Np] ==>
    v(case:[dobj(phys_obj)]), np.
vp(vp(V, Np-D))^[V, Np] ==>
    v(case:[dobj(D)]), { D \== phys_obj }, np.      % RES

%   for move.out2, RES
% vp(vp(V, Np-D, part(P)-index(N)))^[V, Np, _Adv]
%    ==> v(case:[vpart(P),dobj(D),index(N)]), np, adv.
%   for move.out2, RES
% vp(vp(V, Np-D, part(P)-index(N), Rvp))^[V, Np, _Adv, Rvp]
%    ==> v(case:[vpart(P),dobj(D),index(N)]), np, adv, rvp.
%   for move.out4, RES

vp(vp(v(Vaux, v(V, part(P))), Np-D))^[Vaux, V, Np, _Adv] ==>
    vaux, v(case:[vpart(P),dobj(D)]), np, adv(P).

%   for move.out4, RES
vp(vp(v(Vaux, v(V, part(P))), Np-D, Rvp))^[Vaux, V, Np, _Adv, Rvp] ==>
    vaux, v(case:[vpart(P),dobj(D)]), np, adv(P), rvp.

%%% For handling "seat rockets are firing"

vp(vp(v(A, V)))^[A, V] ==>
    be, vf.
```

```
%   be, vf(case:[vcase(state)]).

%   RES, 12/7/89, for "... reported"
vp(case:[dobj(D)])+vtype(T):0.95 ==> v(case:[dobj(D),vtype(T)]).

%   RES, 12/12/89, for "... get set ..."
vp(vp(V-vtype(T), idiomad(Adv)))^[V, Adv] ==>
    v(case:[idiomad(A),vtype(T)]), adv(A).

vp(vp(V, Rvp-default(P)))^[V, Rvp] ==>
    v(case:[vprep(P)]), rvp.
vp:0.90 ==> v.
vp:0.96 ==> v,infp,rvp.                 %rsb - 4-100:12:05:35
vp:0.76 ==> v, rvp.
vp:0.85 ==> v, prpt, rvp.        %rsb - "start looking for ..."
vp:0.83 ==> do, neg, infc.       %rsb - "I didn't know where ..."
vp:0.70 ==> v, crvp, conj, crvp.
vp:0.88 ==> v, rvp, degrs.
vp:0.84 ==> v, rvp, spec.
vp:0.80 ==> v, rvp, spec, degrs.
vp:0.78 ==> auxp, cvp, conj, cvp.
vp:0.72 ==> cvp, conj, cvp.

%%% ---------------  Question Verb Phrase  ---------------

qvp:0.95 ==> do, np, infin.      %rsb - e.g. "did you fire"
qvp:0.95 ==> mod, np, infin.            %rsb - e.g. "can you fire"
qvp ==> do, np, infin, rvp.

%   RES, 12/7/89, for "can you see it"
qvp(qvp(Mod, Np, Inf-vtype(T), Rvp-vcase(theme)))^[Mod, Np, Inf, Rvp] ==>
    mod, np, infin(case:[dobj(phys_obj),vtype(T)]), rvp.

qvp:0.90 ==> mod, np, infin, rvp.
qvp ==> auxp, np, vf, rvp.          %rsb - e.g. ?
qvp:0.85 ==> auxp, np, rvp.      %rsb - e.g. 4-100:13:56:13
qvp:0.80 ==> auxp, np.                  %rsb

%%% ---------------  Imperative Verb Phrase  ---------------

%   RES, 12/8/89, for "get ... NP ... out"
ivp(ivp(imph(Infin, part(Adv))-vtype(T), rvp(Np-vcase(theme))))^
        [Infin,Np,Adv] ==>
    infin(case:[vpart(P),dobj(phys_obj),vtype(T)]), np, adv(P).

%   RES, 12/7/89, for "finish your repositioning"
ivp(ivp(Imph-vtype(T), Rvp-vcase(theme)))^[Imph, Rvp] ==>
    imph(case:[dobj(phys_obj),vtype(T)]), rvp.

ivp:0.90 ==> imph, rvp.
%   for "button up", RES
ivp(ivp(Imph, part(Adv)))^[Imph, adv(Adv)] ==> imph(case:[vpart(P)]), adv(P).
ivp ==> imph.                    %rsb - e.g. 4-100:09:09:18
ivp ==> am, ivp.                 %rsb - 2-100:8:42:27
ivp ==> let, pronoun, vp.        %rsb - "let's go"

%%% ---------------  Imperative  ---------------

%   RES, 12/12/89, for "... start (looking for ) ..."
imph+vtype(T) ==> infin(case:[dobj(infinphr),vtype(T)]).
```

```
imph:0.90 ==> infin.
imph ==> be, ppt.

%   RES, 12/12/89, for "... start (looking for ) ..."
imph(imph(Infin-vtype(T1), Prpt-vtype(T2), Rvp-vcase(theme)))^
        [Infin,Prpt,Rvp] ==>
    infin(case:[dobj(infinphr),vtype(T1)]),
    prpt(case:[dobj(phys_obj),vtype(T2)]),
    rvp.

imph:0.90 ==> infin, prpt, rvp.         %rsb - see next ex.
imph ==> infin, prpt.                   %rsb - "start shooting"
imph ==> be, adj.               %rsb - e.g. 4-100:12:52:40

%%% ---------------  Conjoined Verb Phrase  ---------------

cvp:1.00 ==> v.
cvp:0.96 ==> v, rvp.
cvp:0.85 ==> v, prespart, rvp.          %rsb - 2-100:8:36:56, etc.

%%% ---------------  Verbs  ---------------

v ==> vaux, v.
v(case:X):0.90 ==> v(case:[vprep(P)|X]), prep(P).         % RES

%   for move.out1
% v(v(V, Np-D, part(Adv)))^[V, Np, adv(Adv)] ==>          % RES
%    v(case:[vpart(P),dobj(D)]), np, adv(P).
%   for move.out2, RES
% v(case:[vpart(P),dobj(D),index(1)])+index(1):0.85 ==>
%    infin(case:[vpart(P),dobj(D)]).
%   for move.out3, RES
% v(v(Vaux, v(V, part(P)), Np-D))^[Vaux, V, Np, _Adv] ==>
%    vaux, v(case:[vpart(P),dobj(D)]), np, adv(P).

%   RES, 12/12/89, for "... get set ..."
v(v(Infin-vtype(T), idiomad(Adv)))^[Infin, Adv] ==>
    infin(case:[idiomad(A),vtype(T)]), adv(A).

v:0.90 ==> infin.
v:1.00 ==> mod, av, auxp, av, vf.
v:0.96 ==> mod, av, auxp, vf.
v:0.92 ==> mod, av, infin.
v:0.88 ==> mod, auxp, av, vf.
v:0.84 ==> mod, auxp, vf.
v:0.80 ==> mod, infin.
v:0.60 ==> mod, av, auxp.       %rsb
v:0.50 ==> mod, auxp.                   %rsb - ellipsis ("it must be)

%   for "shot", RES,
%  pastpart with passive voice-case -> tag 'v' with 'passive'
% v(case:X)+passive:0.90 ==> pastpart(case:[voice(passive)|X]).

v(case:X)+passive ==> pastpart(case:[voice(passive)|X]).
v:0.90 ==> past.
v:0.90 ==> be.
v:0.80 ==> be, np, av.              %ejg - "missile is several feet out"
v:0.70 ==> be, np.          %ejg - "missile is several feet"
v:0.70 ==> be, av.

v:1.00 ==> av, auxp, av, vf.
```

```
v:1.00 ==> av, auxp, av, gon, infin.       %rsb - see next rule
v:0.96 ==> av, auxp, gon, infin.    %rsb - see next rule for example
v:0.96 ==> auxp, av, gon, infin.       %rsb - e.g. 4-100:09:09:18
v:0.92 ==> auxp, gon, infin.              %rsb - see above rule
v:0.96 ==> av, auxp, vf.
v:0.92 ==> av, infin.
v:1.00 ==> auxp, av, vf.
v:1.00 ==> auxp, vf.

%v(v(Neg, V)-vtype(T))^[Neg, V] ==>
%   neg(N), ppt(case:[vneg(N), vtype(T)]).

v:1.00 ==> negmod, auxp, av, vf.
v:0.90 ==> negmod, auxp, vf.
v:0.80 ==> negmod, infin.

v ==> do, neg, infin.
v ==> do, infin.
v ==> be, spec.
v ==> negmod, be.
v ==> pres.
v ==> av, pres.

%%% ---------------  Verb Auxiliary  ---------------

vaux ==> mod.

%%% ---------------  Auxiliary Phrase  ---------------

auxp:1.00 ==> have, av, be.
auxp:1.00 ==> have, be.
auxp:0.93 ==> have, av.
auxp:0.80 ==> have.

auxp:1.00 ==> be, av, be.
auxp:0.93 ==> be, av.
auxp:1.00 ==> be, be.                   %rsb - changed weight from .86 to 1.
auxp:0.80 ==> be.

%%% ---------------  Verb Form  ---------------

vf ==> prpt.
vf ==> ppt.
vf ==> cppt, conj, cppt.

%%% ---------------  Past Participle Phrase  ---------------

ppt ==> pastpart.

%%% ---------------  Conjoined Past Participle Phrase  ---------------

cppt ==> ppt.

%%% ---------------  Present Particple Phrase  ---------------

%   RES, 12/12/89, for "... looking for ..."
prpt(case:X) ==> prespart(case:[vprep(P)|X]), prep(P).

%   RES, 12/12/89, was 1.00
prpt:0.90 ==> prespart.
```

```
%%% --------------- Infinitival Clause ---------------

infc:1.00 ==> infp, rvp.
infc:0.80 ==> infp.

%%% --------------- Infinitival Phrase ---------------

infp:1.00 ==> to, be, ppt.
infp:0.96 ==> to, be, prpt.      %rsb - e.g. 4-100:13:56:55
infp:0.85 ==> to, be.                   %rsb - e.g. "he's supposed to be with 85"
infp:0.80 ==> to, adv, infin.
infp:0.80 ==> to, infin.

%%% --------------- Conjoined Infinitival Clause ---------------

cinfc:1.00 ==> conj, infc.
cinfc:0.90 ==> conj, infin.
cinfc:0.80 ==> conj, infin, rvp.

%%% --------------- Relative Verb Phrase ---------------

rvp:1.00 ==> adjp, am.
rvp:0.80 ==> adjp.
rvp:0.75 ==> crvp, conj, crvp.      %rsb - 2:100:9:1:54
rvp:0.90 ==> np.
rvp:1.00 ==> reflxpro.              %rsb - reflexives
%rvp:0.90 ==> np, am.
rvp:0.87 ==> np, am.
rvp:0.90 ==> reflxpro, am.   %rsb
rvp:0.80 ==> np, infc.
rvp:0.85 ==> reflxpro, am.   %rsb
rvp:0.60 ==> np, np.         %rsb - for bitransitives (e.g. 4-100:09:54:32)
rvp ==> am.          %res - "... get set ..."
%rvp:0.90 ==> am.            %res - "... get set ..."
rvp:0.85 ==> infp, np, am.
rvp:0.80 ==> am, np, am.
rvp:0.75 ==> am, np.
rvp:0.70 ==> am, adjp, am.
rvp:0.65 ==> am, infc.
rvp:0.80 ==> sntcomp.               %rsb - changed comp to sntcomp
rvp:0.80 ==> np, sntcomp.
rvp:1.00 ==> infc, cinfc.
rvp:0.80 ==> infc.

%%% --------------- Conjoined Relative Phrase ---------------

crvp ==> np.
crvp:0.90 ==> adjp.              %rsb - 2-100:9:1:54
crvp:0.85 ==> am.                %rsb - same as above

%%% --------------- Sentential Complement ---------------

sntcomp ==> comp, declarative.          %rsb - e.g. 4-100:13:58:11
sntcomp:0.95 ==> declarative.           %rsb - see above rule

%%% --------------- Adverbial Modifier ---------------

am(am(Tree))^[Sub] ==> avph, {Tree =.. [avp|Sub]}.

%%% --------------- Adverbial Phrase ---------------
```

```
% here's a good example of how to get the same effect as the older grammars
% did with trees. Here we explicitly build the trees with lists, and then
% turn the list into a structure in the calling node (am). Complex, yes,
% but if you want to do this, now you know how.

avph([Tp|Avph])^[Tp, Avph]:1.00 ==> tp, avph.
avph([Tp])^[Tp]:1.00 ==> tp.
avph([Tp, Ctp|Avph])^[Tp, Ctp, Avph]:0.86 ==> tp, ctp, avph.
avph([Tp, Ctp])^[Tp, Ctp]:0.80 ==> tp, ctp.
avph([Adv,Pp])^[Adv, Pp]:1.00 ==> adv, pp. %rsb - e.g."right off the road"
avph([Pp|Avph])^[Pp, Avph]:0.90 ==> pp, avph.
avph([Pp])^[Pp]:0.89 ==> pp.                        %increased weight form .83
avph([Pp, Cpp|Avph])^[Pp, Cpp, Avph]:0.76 ==> pp, cpp, avph.
avph([Pp, Cpp])^[Pp, Cpp]:0.70 ==> pp, cpp.
avph([Adv|Avph])^[Adv, Avph]:1.00 ==> adv, avph.
avph([Adv])^[Adv]:0.95 ==> adv.
avph([Evaladv|Avph])^[Evaladv, Avph]:0.90 ==> evaladv, avph.
avph([Evaladv])^[Evaladv]:0.85 ==> evaladv.
avph([Ac])^[Ac]:0.80 ==> ac.
avph([Av])^[Av]:1.00 ==> av.                         %rsb - to access av, avh rules

%%% ---------------  Adverbial ---------------

av:1.00 ==> avh, av.
av:0.80 ==> avh.

%%% ---------------  Adverbial Head  ---------------

avh ==> adv.
avh ==> evaladv.
avh ==> numadv.
avh ==> timadv.
avh ==> demadv.
avh ==> neg.
avh ==> quantadv.                  %rsb
avh ==> ordinal.                   %rsb - "delta will jump first"

%%% ---------------  Adverbial Clause  ---------------

ac ==> subconj, declarative.
ac ==> interpro, declarative.          %rsb for headless rc's and embedded questions
ac ==> subconj, adj.
ac ==> subconj, ajc.
ac:0.85 ==> prep, interpro, declarative.     %rsb "be ready for when he pops out"
ac:0.85 ==> prep, ajc.                     %ejg "prepare target for hoisting"

%%% ---------------  Prepositional Phrase  ---------------

pp ==> prepmod, prep, np.
pp ==> prepmod, prep, prep, np.
pp ==> prep, cnp, comma, cnp, comma, cnp, comma, conj, cnp.
pp ==> prep, cnp, comma, cnp, comma, conj, cnp.
pp ==> prep, cnp, conj, cnp.
pp ==> prep, np.
pp ==> prep, ordseq.

%%% ---------------  Conjoined Prepositional Phrase  ---------------

cpp ==> conj, pp.
cpp ==> disj, pp.
```

```
%%% ---------------  Time Phrase  ---------------

tp ==> timadv.
tp ==> bprep, time, conj, time.
tp ==> prep, time, hours.
tp ==> prep, time.

%%% ---------------  Conjoined Time Phrase  ---------------

ctp ==> conj, tp.
ctp ==> disj, tp.

%%% ---------------  Time  ---------------

time ==> ztime(T),
         { ( ( integer(T), Time = T )
           | ( atom(T),
               atom_chars(T, Tchars),
               number_chars(Time, Tchars) )
           ),
           Time < 2400 }.

time ==> integer(T),
         { ( ( integer(T), Time = T )
           | ( atom(T),
               atom_chars(T, Tchars),
               number_chars(Time, Tchars) )
           ),
           Time < 2400 }.

%%% ---------------  Date  ---------------

date ==> month, day, comma, year.
day ==> integer.
year ==>integer.

%%% ===============  LEXCAT rules  ===============

% we now have yet another lexicalization scheme. For simplicity at this
% point, we just hand-code dictionary facts as
%   dict(Cat, Base, Prs, Args, Tree, Weight)
% where Base, Prs, and Tree can be _, but the rest should be ground
% all tree fiddling happens in all_dict, in bupp_parse, based on
%   tree_swap(Cat, Features)
% facts, if any (but since you have Tree here, you can restrict it
% for various purposes)

tree_swap(noun, [intnoun, month, ordnoun, spac, temp, unit, wsj, dir,
        loc, view, verbal]).

% took out:  veh, vis, obj, acty, subst, agt, type, eqt, ev, fig.

dict(adj, _Word, _Prs, [], _Tree, 1.0).
dict(adv, Word, _Prs, [Word], _Tree, 1.0).   % for case frame
dict(adv, _Word, _Prs, [], _Tree, 1.0).
dict(art, _Word, _Prs, [], _Tree, 1.0).
dict(be, _Word, _Prs, [], _Tree, 1.0).
dict(bprep, _Word, _Prs, [], _Tree, 1.0).
dict(by, _Word, _Prs, [], _Tree, 1.0).
dict(callsign, _Word, _Prs, [], _Tree, 1.0).
dict(cardinal, _Word, _Prs, [], _Tree, 1.0).
```

```
dict(colon, _Word, _Prs, [], _Tree, 1.0).
dict(comp, _Word, _Prs, [], _Tree, 1.0).
dict(comma, _Word, _Prs, [], _Tree, 1.0).
dict(conj, _Word, _Prs, [], _Tree, 1.0).
dict(com, _Word, _Prs, [], _Tree, 1.0).        %changed conv to com
dict(coord, _Word, _Prs, [], _Tree, 1.0).
dict(correc, _Word, _Prs, [], _Tree, 1.0).
dict(demadv, _Word, _Prs, [], _Tree, 1.0).
dict(dempro, _Word, _Prs, [], _Tree, 1.0).
dict(dim, _Word, _Prs, [], _Tree, 1.0).
dict(dir, _Word, _Prs, [], _Tree, 1.0).
dict(diradj, _Word, _Prs, [], _Tree, 1.0).
dict(disj, _Word, _Prs, [], _Tree, 1.0).
dict(do, _Word, _Prs, [], _Tree, 1.0).
dict(evaladj, _Word, _Prs, [], _Tree, 1.0).
dict(evaladv, _Word, _Prs, [], _Tree, 1.0).
dict(exp, _Word, _Prs, [], _Tree, 1.0).
dict(float, _Word, _Prs, [], _Tree, 1.0).
dict(gen, _Word, _Prs, [], _Tree, 1.0).
dict(gon, _Word, _Prs, [], _Tree, 1.0).
dict(grid, _Word, _Prs, [], _Tree, 1.0).
dict(hours, _Word, _Prs, [], _Tree, 1.0).
dict(infin, _Word, _Prs, [], _Tree, 1.0).
dict(integer,  N, _Prs, [N], _Tree, 1.0).
dict(integer, _Word, _Prs, [], _Tree, 1.0).
dict(interpro, _Word, _Prs, [], _Tree, 1.0).
dict(intro, _Word, _Prs, [], _Tree, 1.0).
dict(have, _Word, _Prs, [], _Tree, 1.0).
dict(let, _Word, _Prs, [], _Tree, 1.0).        %rsb - "let pron vp"
dict(letter, _Word, _Prs, [], Tree, 1.0).   %rsb - letter -> letter letter
dict(loc, _Word, _Prs, [], _Tree, 1.0).
dict(lparen, _Word, _Prs, [], _Tree, 1.0).
dict(mod, _Word, _Prs, [], _Tree, 1.0).
dict(month, _Word, _Prs, [], _Tree, 1.0).
dict(neg, _Word, _Prs, [], _Tree, 1.0).
dict(negmod, _Word, _Prs, [], _Tree, 1.0).
dict(noun, _Word, Prs, [F], Tree, 1.0) :-
   member(F, Prs),
   functor(Tree, F, _).  % enforce agreement between functor of Tree and F
dict(noun, _Word, _Prs, [], _Tree, 1.0).
dict(numadj, _Word, _Prs, [], _Tree, 1.0).
dict(numadv, _Word, _Prs, [], _Tree, 1.0).
dict(of, _Word, _Prs, [], _Tree, 1.0).
dict(ordinal, _Word, _Prs, [], _Tree, 1.0).
dict(order, _Word, _Prs, [], _Tree, 1.0).
dict(own, _Word, _Prs, [], _Tree, 1.0).
dict(past, _Word, _Prs, [], _Tree, 1.0).
dict(pastpart, _Word, _Prs, [], _Tree, 1.0).
dict(posnoun, _Word, _Prs, [], _Tree, 1.0).
dict(pospro, _Word, _Prs, [], _Tree, 1.0).
dict(prep, Word, _Prs, [Word], _Tree, 1.0).   % for case frame
dict(prep, _Word, _Prs, [], _Tree, 1.0).
dict(prepmod, _Word, _Prs, [], _Tree, 1.0).
dict(pres, _Word, _Prs, [], _Tree, 1.0).
dict(prespart, _Word, _Prs, [], _Tree, 1.0).
dict(pronoun, _Word, _Prs, [], _Tree, 1.0).
dict(quant, _Word, _Prs, [], _Tree, 1.0).
dict(quantadv, _Word, _Prs, [], _Tree, 1.0).
dict(quantpro, _Word, _Prs, [], _Tree, 1.0).
dict(quote, _Word, _Prs, [], _Tree, 1.0).
dict(refadj, _Word, _Prs, [], _Tree, 1.0).
```

```
dict(reflxpro, _Word, _Prs, [], _Tree, 1.0).
dict(relpro, _Word, _Prs, [], _Tree, 1.0).
dict(reqadv, _Word, _Prs, [], _Tree, 1.0).
dict(resp, _Word, _Prs, [], _Tree, 1.0).
dict(rparen, _Word, _Prs, [], _Tree, 1.0).
dict(spec, _Word, _Prs, [], _Tree, 1.0).
dict(subconj, _Word, _Prs, [], _Tree, 1.0).
dict(that, _Word, _Prs, [], _Tree, 1.0).
dict(timadj, _Word, _Prs, [], _Tree, 1.0).
dict(timadv, _Word, _Prs, [], _Tree, 1.0).
dict(to, _Word, _Prs, [], _Tree, 1.0).
dict(tprep, _Word, _Prs, [], _Tree, 1.0).
dict(view, _Word, _Prs, [], _Tree, 1.0).
dict(verbal, _Word, _Prs, [], _Tree, 1.0).
dict(ztime,  N, _Prs, [N], _Tree, 1.0).
dict(ztime, _Word, _Prs, [], _Tree, 1.0).

%%% LEXCATWORD rules needed

dict(punct,  Word, _Prs, [Word], _Tree, 1.0).
% '-' - tprep
dict(tprep,  Word, _Prs, [Word], _Tree, 1.0).


/* lxitrans(+Cat, -Substitute)
-  a simple substitution for certain categories. */

lxitrans(plural, noun).
lxitrans(sing, noun).
lxitrans(pastpart, ppt).
lxitrans(prespart, prpt).
lxitrans(infin, pres).
lxitrans(third_pres, pres).
```

APPENDIX E

TYPE HIERARCHY

```
:- dynamic a_kind_of/5.
:- dynamic root/3.

/* sccs('%W% %G%'). */

root(e_r_concept, [], 1).

%%% ***************  CLASS Definitions  ***************

a_kind_of(e_r_concept, root-1, [], _, []).
a_kind_of(obj_concept, noun-1, e_r_concept, _, []).
a_kind_of(act_concept, infin-1, e_r_concept, _, []).
a_kind_of(phys_obj, noun-1, obj_concept, noun-1, []).
a_kind_of(location, noun-1, obj_concept, noun-1, []).
a_kind_of(abst_obj, noun-1, obj_concept, noun-1, []).
a_kind_of(stative_act, infin-1, act_concept, infin-1, []).
a_kind_of(event_act, infin-1, act_concept, infin-1, []).

a_kind_of('A-3', noun-1, 'Skywarrior', noun-1, []).
a_kind_of('A-3', noun-1, program, noun-1, []).
a_kind_of('A-3B', noun-1, 'A-3', noun-1, []).
a_kind_of('A-4', noun-1, 'Skyhawk 2', noun-1, []).
a_kind_of('A-4B', noun-1, 'A-4', noun-1, []).
a_kind_of('A-4C', noun-1, 'A-4', noun-1, []).
a_kind_of('A-4E', noun-1, 'A-4', noun-1, []).
a_kind_of('A-4M', noun-1, 'A-4', noun-1, []).
a_kind_of('A-5', noun-1, 'bomber aircraft', noun-1, []).
a_kind_of('A-5', noun-1, program, noun-1, []).
a_kind_of('A-5C', noun-1, 'A-5', noun-1, []).
a_kind_of('A-6', noun-1, 'bomber aircraft', noun-1, []).
a_kind_of('A-6', noun-1, program, noun-1, []).
a_kind_of('A-6A', noun-1, 'A-6', noun-1, []).
a_kind_of('A-6B', noun-1, 'A-6', noun-1, []).
a_kind_of('A-6E', noun-1, 'A-6', noun-1, []).
a_kind_of('A-7', noun-1, 'attack aircraft', noun-1, []).
a_kind_of('A-7', noun-1, program, noun-1, []).
a_kind_of('A-7B/E', noun-1, 'A-7', noun-1, []).
a_kind_of('A-7C', noun-1, 'A-7', noun-1, []).
a_kind_of('A-7E', noun-1, 'A-7', noun-1, []).
a_kind_of('AAR-47', noun-1, 'electro-optical device', noun-1, []).
a_kind_of('AC-130', noun-1, 'cargo aircraft', noun-1, []).
a_kind_of('AC-130A', noun-1, 'AC-130', noun-1, []).
a_kind_of('ACIMD', noun-1, 'air-to-air missile', noun-1, []).
a_kind_of('ACIMD', noun-1, program, noun-1, []).
a_kind_of('AFR', noun-1, organization, noun-1, []).
a_kind_of('AGM-123', noun-1, 'Skipper', noun-1, []).
a_kind_of('AGM-123A', noun-1, 'AGM-123', noun-1, []).
a_kind_of('AGM-45', noun-1, 'Shrike', noun-1, []).
a_kind_of('AGM-65', noun-1, 'Maverick', noun-1, []).
a_kind_of('AGM-65C', noun-1, 'AGM-65', noun-1, []).
a_kind_of('AGM-65D', noun-1, 'AGM-65', noun-1, []).
a_kind_of('AGM-65E', noun-1, 'AGM-65', noun-1, []).
a_kind_of('AGM-88', noun-1, 'Harm', noun-1, []).
```

```
a_kind_of('AGM-88I', noun-1, 'AGM-88', noun-1, []).
a_kind_of('AH-1', noun-1, 'military helicopter', noun-1, []).
a_kind_of('AH-1J', noun-1, 'AH-1', noun-1, []).
a_kind_of('AH-1W', noun-1, 'Super Cobra', noun-1, []).
a_kind_of('AIM-54', noun-1, 'Phoenix', noun-1, []).
a_kind_of('AIM-54C', noun-1, 'AIM-54', noun-1, []).
a_kind_of('AIM-9', noun-1, 'Sidewinder', noun-1, []).
a_kind_of('AIM-9B', noun-1, 'AIM-9', noun-1, []).
a_kind_of('AIM-9C', noun-1, 'AIM-9', noun-1, []).
a_kind_of('AIM-9D', noun-1, 'AIM-9', noun-1, []).
a_kind_of('AIM-9L', noun-1, 'AIM-9', noun-1, []).
a_kind_of('AIM-9M', noun-1, 'AIM-9', noun-1, []).
a_kind_of('AIM-9R', noun-1, 'AIM-9', noun-1, []).
a_kind_of('AJ', noun-1, id, noun-1, []).
a_kind_of('ALQ-119', noun-1, 'ECM pod', noun-1, []).
a_kind_of('ASROC engine', noun-1, 'solid propellant rocket engine', noun-1, []).
a_kind_of('ATB', noun-1, 'bomber aircraft', noun-1, []).
a_kind_of('ATB', noun-1, program, noun-1, []).
a_kind_of('AV-8', noun-1, 'Harrier', noun-1, []).
a_kind_of('AV-8', noun-1, program, noun-1, []).
a_kind_of('AV-8A', noun-1, 'AV-8', noun-1, []).
a_kind_of('AV-8B', noun-1, 'AV-8', noun-1, []).
a_kind_of('Adam', noun-1, 'search set pod', noun-1, []).
a_kind_of('Aero 51A', noun-1, trailer, noun-1, []).
a_kind_of('Agile-Quickturn', noun-1, missile, noun-1, []).
a_kind_of('Aircraft Survivability Laboratory', noun-1, laboratory, noun-1, []).
a_kind_of('Antonio', noun-1, 'human being', noun-1, []).
a_kind_of('April', noun-1, month, noun-1, []).
a_kind_of('Argus Range', noun-1, 'bombing range', noun-1, []).
a_kind_of('Armitage Field', noun-1, airport, noun-1, []).
a_kind_of('Assault Breaker', noun-1, sled, noun-1, []).
a_kind_of('B-1-B', noun-1, target, noun-1, []).
a_kind_of('B-2', noun-1, 'ATB', noun-1, []).
a_kind_of('B-61', noun-1, bomb, noun-1, []).
a_kind_of('B57-VFA-82A', noun-1, 'camera pod', noun-1, []).
a_kind_of('BLU-80', noun-1, 'Bigeye', noun-1, []).
a_kind_of('BLU-80/B', noun-1, 'BLU-80', noun-1, []).
a_kind_of('BLU-95', noun-1, 'FAE', noun-1, []).
a_kind_of('BLU-96', noun-1, 'FAE', noun-1, []).
a_kind_of('BQM', noun-1, drone, noun-1, []).
a_kind_of('BQM-34S', noun-1, 'BQM', noun-1, []).
a_kind_of('BTV', noun-1, 'test vehicle', noun-1, []).
a_kind_of('BTV', noun-1, 'Walleye', noun-1, []).
a_kind_of('Bat', noun-1, 'air-to-surface missile', noun-1, []).
a_kind_of('Bat', noun-1, program, noun-1, []).
a_kind_of('Beavertail cactus', noun-1, wildflower, noun-1, []).
a_kind_of('Beling', noun-1, 'human being', noun-1, []).
a_kind_of('Benjes', noun-1, 'human being', noun-1, []).
a_kind_of('Bigeye', noun-1, bomb, noun-1, []).
a_kind_of('Bigeye', noun-1, program, noun-1, []).
a_kind_of('Bullpup', noun-1, 'air-to-surface missile', noun-1, []).
a_kind_of('Bullpup', noun-1, program, noun-1, []).
a_kind_of('C-130', noun-1, 'cargo aircraft', noun-1, []).
a_kind_of('C-130A', noun-1, 'C-130', noun-1, []).
a_kind_of('CFT', noun-1, test, noun-1, []).
a_kind_of('CL', noun-1, id, noun-1, []).
a_kind_of('CLAM B', noun-1, 'Fireye', noun-1, []).
a_kind_of('CSWS dispenser', noun-1, dispenser, noun-1, []).
a_kind_of('CVA-16', noun-1, 'USS Lexington', noun-1, []).
a_kind_of('CVA-59', noun-1, 'USS Forrestal', noun-1, []).
a_kind_of('Canada', noun-1, country, noun-1, []).
```

```
a_kind_of('Captain', noun-1, rank, noun-1, []).
a_kind_of('China Lake', noun-1, lake, noun-1, []).
a_kind_of('China Lake', noun-1, id, noun-1, []).
a_kind_of('Code', noun-1, organization, noun-1, []).
a_kind_of('Cold Line', noun-1, land, noun-1, []).
a_kind_of('Commander', noun-1, rank, noun-1, []).
a_kind_of('Condor', noun-1, 'air-to-surface missile', noun-1, []).
a_kind_of('Condor', noun-1, program, noun-1, []).
a_kind_of('Coso Range', noun-1, 'bombing range', noun-1, []).
a_kind_of('DD-851', noun-1, 'USS Repertus', noun-1, []).
a_kind_of('DODX', noun-1, boxcar, noun-1, []).
a_kind_of('DSU-28', noun-1, 'TDD', noun-1, []).
a_kind_of('DSU-28B', noun-1, 'DSU-28', noun-1, []).
a_kind_of('DVT-2R', noun-1, 'escape system', noun-1, []).
a_kind_of('DVT-7', noun-1, 'escape system', noun-1, []).
a_kind_of('Dalea', noun-1, wildflower, noun-1, []).
a_kind_of('Dev-Assist', noun-1, equipment, noun-1, []).
a_kind_of('ECM pod', noun-1, pod, noun-1, []).
a_kind_of('EDM-4', noun-1, engine, noun-1, []).
a_kind_of('ESS', noun-1, simulator, noun-1, []).
a_kind_of('EX-62', noun-1, 'TDD', noun-1, []).
a_kind_of('Eglin AFB', noun-1, organization, noun-1, []).
a_kind_of('Eglin AFB', noun-1, location, noun-1, []).
a_kind_of('F-14', noun-1, 'Tomcat', noun-1, []).
a_kind_of('F-14A', noun-1, 'Tomcat', noun-1, []).
a_kind_of('F-15', noun-1, 'fighter aircraft', noun-1, []).
a_kind_of('F-15', noun-1, program, noun-1, []).
a_kind_of('F-3', noun-1, 'fighter aircraft', noun-1, []).
a_kind_of('F-3', noun-1, program, noun-1, []).
a_kind_of('F-3D', noun-1, 'F-3', noun-1, []).
a_kind_of('F-3D-1', noun-1, 'F-3H', noun-1, []).
a_kind_of('F-3H', noun-1, 'F-3', noun-1, []).
a_kind_of('F-3H-1', noun-1, 'F-3H', noun-1, []).
a_kind_of('F-4', noun-1, 'Wild Weasel', noun-1, []).
a_kind_of('F-4', noun-1, program, noun-1, []).
a_kind_of('F-4B', noun-1, 'F-4', noun-1, []).
a_kind_of('F-4G', noun-1, 'F-4', noun-1, []).
a_kind_of('F-4J', noun-1, 'F-4', noun-1, []).
a_kind_of('F-86', noun-1, 'fighter aircraft', noun-1, []).
a_kind_of('F-86', noun-1, program, noun-1, []).
a_kind_of('F-9', noun-1, 'fighter aircraft', noun-1, []).
a_kind_of('F-9', noun-1, program, noun-1, []).
a_kind_of('F/A-18', noun-1, 'Hornet', noun-1, []).
a_kind_of('F/A-18', noun-1, program, noun-1, []).
a_kind_of('F/A-18 simulator', noun-1, 'cockpit simulator', noun-1, []).
a_kind_of('F/A-18A', noun-1, 'F/A-18', noun-1, []).
a_kind_of('F/A-18A simulator', noun-1, 'F/A-18 simulator', noun-1, []).
a_kind_of('F/A-18A Validation Laboratory', noun-1, room, noun-1, []).
a_kind_of('F/A-18C', noun-1, 'F/A-18', noun-1, []).
a_kind_of('FAE', noun-1, bomb, noun-1, []).
a_kind_of('FBM', noun-1, 'surface-to-surface missile', noun-1, []).
a_kind_of('FBM', noun-1, program, noun-1, []).
a_kind_of('FLIR detector', noun-1, 'infrared detector', noun-1, []).
a_kind_of('FSQ-12', noun-1, 'VLA', noun-1, []).
a_kind_of('FTV-2', noun-1, 'Agile-Quickturn', noun-1, []).
a_kind_of('Fireye', noun-1, bomb, noun-1, []).
a_kind_of('Fireye', noun-1, program, noun-1, []).
a_kind_of('Florida', noun-1, land, noun-1, []).
a_kind_of('Fulmer', noun-1, 'human being', noun-1, []).
a_kind_of('G Range', noun-1, 'missile range', noun-1, []).
a_kind_of('G-1 Range', noun-1, 'missile range', noun-1, []).
```

```
a_kind_of('G-2 Range', noun-1, 'missile range', noun-1, []).
a_kind_of('G-R simulator', noun-1, simulator, noun-1, []).
a_kind_of('GBU-22', noun-1, 'LLLGB', noun-1, []).
a_kind_of('GBU-22B', noun-1, 'GBU-22', noun-1, []).
a_kind_of('Gallinetti', noun-1, 'human being', noun-1, []).
a_kind_of('Gatling gun', noun-1, gun, noun-1, []).
a_kind_of('Goshawk', noun-1, aircraft, noun-1, []).
a_kind_of('Germain', noun-1, 'human being', noun-1, []).
a_kind_of('Gulf of Tonkin', noun-1, gulf, noun-1, []).
a_kind_of('HBX', noun-1, explosive, noun-1, []).
a_kind_of('HH-1', noun-1, 'military helicopter', noun-1, []).
a_kind_of('HH-1K', noun-1, 'HH-1', noun-1, []).
a_kind_of('HS camera', noun-1, camera, noun-1, []).
a_kind_of('HTW', noun-1, weapon, noun-1, []).
a_kind_of('Harm', noun-1, 'air-to-surface missile', noun-1, []).
a_kind_of('Harm', noun-1, program, noun-1, []).
a_kind_of('Harpoon', noun-1, 'air-to-surface missile', noun-1, []).
a_kind_of('Harpoon', noun-1, program, noun-1, []).
a_kind_of('Harrier', noun-1, 'fighter aircraft', noun-1, []).
a_kind_of('Harrier', noun-1, 'V/STOL aircraft', noun-1, []).
a_kind_of('Harvey Field', noun-1, airport, noun-1, []).
a_kind_of('Hessler', noun-1, 'human being', noun-1, []).
a_kind_of('Hipeg', noun-1, gun, noun-1, []).
a_kind_of('Honeywell', noun-1, organization, noun-1, []).
a_kind_of('Hornet', noun-1, 'fighter aircraft', noun-1, []).
a_kind_of('Hot Line', noun-1, land, noun-1, []).
a_kind_of('ITER', noun-1, launcher, noun-1, []).
a_kind_of('J-52', noun-1, 'jet engine', noun-1, []).
a_kind_of('John', noun-1, 'human being', noun-1, []).
a_kind_of('Jon', noun-1, 'human being', noun-1, []).
a_kind_of('Kapernick', noun-1, 'human being', noun-1, []).
a_kind_of('Kennedy', noun-1, 'human being', noun-1, []).
a_kind_of('Kern River', noun-1, river, noun-1, []).
a_kind_of('LCDR', noun-1, rank, noun-1, []).
a_kind_of('LGB', noun-1, bomb, noun-1, []).
a_kind_of('LLLGB', noun-1, bomb, noun-1, []).
a_kind_of('LWIR detector', noun-1, 'infrared detector', noun-1, []).
a_kind_of('Lieutenant', noun-1, rank, noun-1, []).
a_kind_of('Linda', noun-1, 'human being', noun-1, []).
a_kind_of('Lockheed', noun-1, organization, noun-1, []).
a_kind_of('MAD', noun-1, organization, noun-1, []).
a_kind_of('MIG-21', noun-1, 'mig aircraft', noun-1, []).
a_kind_of('MK-4', noun-1, 'Hipeg', noun-1, []).
a_kind_of('MK-81', noun-1, bomb, noun-1, []).
a_kind_of('MK-82', noun-1, bomb, noun-1, []).
a_kind_of('MK-91', noun-1, 'Silver Bullet', noun-1, []).
a_kind_of('Machine Shop', noun-1, room, noun-1, []).
a_kind_of('Major', noun-1, rank, noun-1, []).
a_kind_of('Marines', noun-1, 'USMC', noun-1, []).
a_kind_of('Marines', noun-1, id, noun-1, []).
a_kind_of('Mark-7', noun-1, 'dolly loader', noun-1, []).
a_kind_of('Martin Marietta', noun-1, organization, noun-1, []).
a_kind_of('Maverick', noun-1, 'air-to-surface missile', noun-1, []).
a_kind_of('Maverick', noun-1, program, noun-1, []).
a_kind_of('Maxwell', noun-1, 'human being', noun-1, []).
a_kind_of('Michelson Laboratory', noun-1, laboratory, noun-1, []).
a_kind_of('Michelson', noun-1, 'human being', noun-1, []).
a_kind_of('Missouri River', noun-1, river, noun-1, []).
a_kind_of('Misty 13', noun-1, id, noun-1, []).
a_kind_of('NASA', noun-1, organization, noun-1, []).
a_kind_of('NG', noun-1, id, noun-1, []).
```

```
a_kind_of('Northrop', noun-1, organization, noun-1, []).
a_kind_of('NWC', noun-1, organization, noun-1, []).
a_kind_of('NWC', noun-1, location, noun-1, []).
a_kind_of('Naces', noun-1, organization, noun-1, []).
a_kind_of('OPNAV', noun-1, organization, noun-1, []).
a_kind_of('Olancha Peak', noun-1, mountain, noun-1, []).
a_kind_of('PCH-1', noun-1, 'USS High Point', noun-1, []).
a_kind_of('PMTC', noun-1, organization, noun-1, []).
a_kind_of('PMTC', noun-1, location, noun-1, []).
a_kind_of('PMTC', noun-1, id, noun-1, []).
a_kind_of('Pacific Ocean', noun-1, ocean, noun-1, []).
a_kind_of('Panamint Valley', noun-1, valley, noun-1, []).
a_kind_of('Pax River', noun-1, river, noun-1, []).
a_kind_of('Phoenix', noun-1, 'Standard Arm', noun-1, []).
a_kind_of('Point Mugu', noun-1, land, noun-1, []).
a_kind_of('Point Mugu', noun-1, id, noun-1, []).
a_kind_of('Puget Sound', noun-1, sound, noun-1, []).
a_kind_of('QF-4', noun-1, 'F-4', noun-1, []).
a_kind_of('QF-4', noun-1, drone, noun-1, []).
a_kind_of('QF-4B', noun-1, 'QF-4', noun-1, []).
a_kind_of('QF-4H-1', noun-1, 'QF-4', noun-1, []).
a_kind_of('QF-86', noun-1, 'F-86', noun-1, []).
a_kind_of('QF-86', noun-1, drone, noun-1, []).
a_kind_of('QF-9', noun-1, 'F-9', noun-1, []).
a_kind_of('QF-9', noun-1, drone, noun-1, []).
a_kind_of('QF-9F', noun-1, 'QF-9', noun-1, []).
a_kind_of('R. W. Strongback', noun-1, 'human being', noun-1, []).
a_kind_of('RA-5C', noun-1, 'A-5C', noun-1, []).
a_kind_of('RAPEC', noun-1, program, noun-1, []).
a_kind_of('RCC', noun-1, building, noun-1, []).
a_kind_of('Richard', noun-1, 'human being', noun-1, []).
a_kind_of('Ridgecrest', noun-1, land, noun-1, []).
a_kind_of('Rockeye II', noun-1, 'Rockeye', noun-1, []).
a_kind_of('Rockeye', noun-1, bomb, noun-1, []).
a_kind_of('Rockeye', noun-1, program, noun-1, []).
a_kind_of('Roseville', noun-1, land, noun-1, []).
a_kind_of('Rzeszotko', noun-1, 'human being', noun-1, []).
a_kind_of('S-2', noun-1, 'snow aircraft', noun-1, []).
a_kind_of('S-2A', noun-1, 'S-2', noun-1, []).
a_kind_of('SH', noun-1, id, noun-1, []).
a_kind_of('SRT-6', noun-1, 'escape system', noun-1, []).
a_kind_of('SSG-N1', noun-1, 'guided missile submarine', noun-1, []).
a_kind_of('Samson', noun-1, drone, noun-1, []).
a_kind_of('Searles Lake', noun-1, lake, noun-1, []).
a_kind_of('Seventh Fleet', noun-1, fleet, noun-1, []).
a_kind_of('Shrike', noun-1, 'air-to-surface missile', noun-1, []).
a_kind_of('Shrike', noun-1, program, noun-1, []).
a_kind_of('Sidearm', noun-1, 'air-to-surface missile', noun-1, []).
a_kind_of('Sidearm', noun-1, program, noun-1, []).
a_kind_of('Sidewinder', noun-1, 'air-to-air missile', noun-1, []).
a_kind_of('Sidewinder', noun-1, program, noun-1, []).
a_kind_of('Sidewinder', noun-1, 'antiaircraft missile', noun-1, []).
a_kind_of('Sierra Nevada', noun-1, land, noun-1, []).
a_kind_of('Silver Bullet', noun-1, bomb, noun-1, []).
a_kind_of('Sinkex', noun-1, operation, noun-1, []).
a_kind_of('Skipper', noun-1, 'LGB', noun-1, []).
a_kind_of('Skipper', noun-1, program, noun-1, []).
a_kind_of('Skyhawk', noun-1, 'bomber aircraft', noun-1, []).
a_kind_of('Skyhawk', noun-1, program, noun-1, []).
a_kind_of('Skyhawk 2', noun-1, 'Skyhawk', noun-1, []).
a_kind_of('Skyray', noun-1, 'instrumentation pod', noun-1, []).
```

```
a_kind_of('Skyray', noun-1, program, noun-1, []).
a_kind_of('Skywarrior', noun-1, 'bomber aircraft', noun-1, []).
a_kind_of('Skywarrior', noun-1, program, noun-1, []).
a_kind_of('Snipe', noun-1, missile, noun-1, []).
a_kind_of('Snort', noun-1, 'test range', noun-1, []).
a_kind_of('Sparrow 3', noun-1, 'Sparrow', noun-1, []).
a_kind_of('Sparrow', noun-1, 'air-to-air missile', noun-1, []).
a_kind_of('Sparrow', noun-1, program, noun-1, []).
a_kind_of('Standard Arm II-N', noun-1, 'Standard Arm', noun-1, []).
a_kind_of('Standard Arm', noun-1, 'air-to-surface missile', noun-1, []).
a_kind_of('Standard Arm', noun-1, program, noun-1, []).
a_kind_of('Stephen', noun-1, 'human being', noun-1, []).
a_kind_of('Super Cobra', noun-1, 'military helicopter', noun-1, []).
a_kind_of('T-5', noun-1, tower, noun-1, []).
a_kind_of('T-45A', noun-1, 'Goshawk', noun-1, []).
a_kind_of('TDD', noun-1, detector, noun-1, []).
a_kind_of('TH-1', noun-1, 'military helicopter', noun-1, []).
a_kind_of('TH-1L', noun-1, 'TH-1', noun-1, []).
a_kind_of('TIM', noun-1, equipment, noun-1, []).
a_kind_of('Tacit/Rainbow', noun-1, 'air-to-air missile', noun-1, []).
a_kind_of('Tomahawk', noun-1, 'cruise missile', noun-1, []).
a_kind_of('Tomcat', noun-1, 'fighter aircraft', noun-1, []).
a_kind_of('Tomcat', noun-1, program, noun-1, []).
a_kind_of('Trident', noun-1, 'surface-to-surface missile', noun-1, []).
a_kind_of('Trident', noun-1, program, noun-1, []).
a_kind_of('UH-1', noun-1, 'military helicopter', noun-1, []).
a_kind_of('UH-1E', noun-1, 'UH-1', noun-1, []).
a_kind_of('UH-1N', noun-1, 'UH-1', noun-1, []).
a_kind_of('UH-2', noun-1, 'military helicopter', noun-1, []).
a_kind_of('UH-2A', noun-1, 'UH-2', noun-1, []).
a_kind_of('USAF', noun-1, organization, noun-1, []).
a_kind_of('USAF_no', noun-1, number, noun-1, []).
a_kind_of('USMC', noun-1, organization, noun-1, []).
a_kind_of('USN', noun-1, organization, noun-1, []).
a_kind_of('USS Forrestal', noun-1, 'aircraft carrier', noun-1, []).
a_kind_of('USS High Point', noun-1, 'hydrofoil craft', noun-1, []).
a_kind_of('USS Kitty Hawk', noun-1, 'aircraft carrier', noun-1, []).
a_kind_of('USS Lexington', noun-1, 'aircraft carrier', noun-1, []).
a_kind_of('USS Repertus', noun-1, destroyer, noun-1, []).
a_kind_of('V/STOL aircraft', noun-1, aircraft, noun-1, []).
a_kind_of('VC-3', noun-1, squadron, noun-1, []).
a_kind_of('VLA', noun-1, 'ASROC engine', noun-1, []).
a_kind_of('VMA-513', noun-1, squadron, noun-1, []).
a_kind_of('VX-5', noun-1, squadron, noun-1, []).
a_kind_of('WF', noun-1, id, noun-1, []).
a_kind_of('WSSA', noun-1, building, noun-1, []).
a_kind_of('Walleye II', noun-1, 'Walleye', noun-1, []).
a_kind_of('Walleye', noun-1, bomb, noun-1, []).
a_kind_of('Walleye', noun-1, program, noun-1, []).
a_kind_of('Weapons Survivability Laboratory', noun-1, laboratory, noun-1, []).
a_kind_of('Weston', noun-1, 'human being', noun-1, []).
a_kind_of('Wild Weasel', noun-1, 'fighter aircraft', noun-1, []).
a_kind_of('Wincn', noun-1, 'human being', noun-1, []).
a_kind_of('XAIM-54', noun-1, 'AIM-54', noun-1, []).
a_kind_of('XAIM-54C', noun-1, 'AIM-54C', noun-1, []).
a_kind_of('XE', noun-1, id, noun-1, []).
a_kind_of('XN-01', noun-1, pedestal, noun-1, []).
a_kind_of('air breathing engine', noun-1, engine, noun-1, []).
a_kind_of('air-to-air missile', noun-1, missile, noun-1, []).
a_kind_of('air-to-air view', noun-1, view, noun-1, []).
a_kind_of('air-to-ground view', noun-1, view, noun-1, []).
```

```
a_kind_of('air-to-surface missile', noun-1, missile, noun-1, []).
a_kind_of('aircraft carrier', noun-1, ship, noun-1, []).
a_kind_of('airport tower', noun-1, tower, noun-1, []).
a_kind_of('antiaircraft missile', noun-1, missile, noun-1, []).
a_kind_of('attack aircraft', noun-1, aircraft, noun-1, []).
a_kind_of('ballistic range', noun-1, 'test range', noun-1, []).
a_kind_of('bomb skid', noun-1, stand, noun-1, []).
a_kind_of('bomber aircraft', noun-1, 'attack aircraft', noun-1, []).
a_kind_of('bombing range', noun-1, 'test range', noun-1, []).
a_kind_of('bottom view', noun-1, view, noun-1, []).
a_kind_of('camera pod', noun-1, pod, noun-1, []).
a_kind_of('cargo aircraft', noun-1, 'transport aircraft', noun-1, []).
a_kind_of('centerline station', noun-1, station, noun-1, []).
a_kind_of('chemical element', noun-1, phys_obj, noun-1, []).
a_kind_of('circuit board', noun-1, board, noun-1, []).
a_kind_of('cockpit simulator', noun-1, 'flight simulator', noun-1, []).
a_kind_of('combat vehicle', noun-1, 'surface vehicle', noun-1, []).
a_kind_of('control equipment', noun-1, equipment, noun-1, []).
a_kind_of('cookoff test', noun-1, test, noun-1, []).
a_kind_of('cruise missile', noun-1, 'surface-to-surface missile', noun-1, []).
a_kind_of('data link pod', noun-1, pod, noun-1, []).
a_kind_of('direct power generator', noun-1, 'electric generator', noun-1, []).
a_kind_of('dolly loader', noun-1, loader, noun-1, []).
a_kind_of('dolly loader', noun-1, dolly, noun-1, []).
a_kind_of('drogue parachute', noun-1, parachute, noun-1, []).
a_kind_of('dump test', noun-1, test, noun-1, []).
a_kind_of('electric battery', noun-1, 'electrochemical cell', noun-1, []).
a_kind_of('electric generator', noun-1, generator, noun-1, []).
a_kind_of('electro-optical device', noun-1, equipment, noun-1, []).
a_kind_of('electrochemical cell', noun-1, phys_obj, noun-1, []).
a_kind_of('escape system', noun-1, system, noun-1, [[ejection, noun-1]]).
a_kind_of('explosive booster', noun-1, 'explosive initiator', noun-1, []).
a_kind_of('explosive device', noun-1, device, noun-1, []).
a_kind_of('explosive initiator', noun-1, 'explosive device', noun-1, []).
a_kind_of('external store', noun-1, phys_obj, noun-1, []).
a_kind_of('fiber optic', noun-1, wire, noun-1, []).
a_kind_of('fighter aircraft', noun-1, 'attack aircraft', noun-1, []).
a_kind_of('flight deck', noun-1, deck, noun-1, []).
a_kind_of('flight path', noun-1, path, noun-1, []).
a_kind_of('flight simulator', noun-1, 'training simulator', noun-1, []).
a_kind_of('flight test', noun-1, test, noun-1, []).
a_kind_of('flying eagle', noun-1, marking, noun-1, []).
a_kind_of('flyover test', noun-1, test, noun-1, []).
a_kind_of('frame view', noun-1, view, noun-1, []).
a_kind_of('front view', noun-1, view, noun-1, []).
a_kind_of('gas turbine engine', noun-1, 'air breathing engine', noun-1, []).
a_kind_of('guided missile submarine', noun-1, submarine, noun-1, []).
a_kind_of('handling equipment', noun-1, equipment, noun-1, []).
a_kind_of('hangar bay', noun-1, deck, noun-1, []).
a_kind_of('homing device', noun-1, equipment, noun-1, []).
a_kind_of('human being', noun-1, primate, noun-1, []).
a_kind_of('hydrofoil craft', noun-1, ship, noun-1, []).
a_kind_of('infrared detector', noun-1, 'infrared instrument', noun-1, []).
a_kind_of('infrared detector', noun-1, detector, noun-1, []).
a_kind_of('infrared instrument', noun-1, 'radiation measuring instrument', noun-1,
[]).
a_kind_of('inside view', noun-1, view, noun-1, []).
a_kind_of('instrumentation pod', noun-1, pod, noun-1, []).
a_kind_of('integrated circuit', noun-1, circuit, noun-1, []).
a_kind_of('jet engine', noun-1, 'gas turbine engine', noun-1, []).
a_kind_of('latch assembly', noun-1, assembly, noun-1, []).
```

```
a_kind_of('lighting equipment', noun-1, equipment, noun-1, []).
a_kind_of('load test', noun-1, test, noun-1, []).
a_kind_of('measuring instrument', noun-1, instrument, noun-1, []).
a_kind_of('mercury vapor', noun-1, mercury_metal, noun-1, []).
a_kind_of('mig aircraft', noun-1, 'fighter aircraft', noun-1, []).
a_kind_of('military helicopter', noun-1, helicopter, noun-1, []).
a_kind_of('missile range', noun-1, 'test range', noun-1, []).
a_kind_of('modified AIM-9C', noun-1, 'AIM-9C', noun-1, []).
a_kind_of('modified AIM-9C', noun-1, 'Sidearm', noun-1, []).
a_kind_of('motor vehicle', noun-1, 'surface vehicle', noun-1, []).
a_kind_of('night hawk', noun-1, marking, noun-1, []).
a_kind_of('night vision', noun-1, vision, noun-1, []).
a_kind_of('optical equipment', noun-1, equipment, noun-1, []).
a_kind_of('overall view', noun-1, view, noun-1, []).
a_kind_of('overhead view', noun-1, view, noun-1, []).
a_kind_of('photoelectric generator', noun-1, 'direct power generator', noun-1, []).
a_kind_of('photographic equipment', noun-1, equipment, noun-1, []).
a_kind_of('photovoltaic cell', noun-1, 'photoelectric generator', noun-1, []).
a_kind_of('pusher sled', noun-1, sled, noun-1, []).
a_kind_of('radiation measuring instrument', noun-1, 'measuring instrument', noun-1,
[]).
a_kind_of('rear view', noun-1, view, noun-1, []).
a_kind_of('rocket burn', noun-1, burn, noun-1, []).
a_kind_of('rocket engine', noun-1, engine, noun-1, []).
a_kind_of('rotary wing aircraft', noun-1, 'V/STOL aircraft', noun-1, []).
a_kind_of('rotary wing dispenser', noun-1, dispenser, noun-1, []).
a_kind_of('rotary wing', noun-1, wing, noun-1, []).
a_kind_of('sea launch', noun-1, launch, noun-1, []).
a_kind_of('search set pod', noun-1, pod, noun-1, []).
a_kind_of('separation test', noun-1, test, noun-1, []).
a_kind_of('side view', noun-1, view, noun-1, []).
a_kind_of('sled test', noun-1, test, noun-1, []).
a_kind_of('snow aircraft', noun-1, aircraft, noun-1, []).
a_kind_of('solder board', noun-1, board, noun-1, []).
a_kind_of('solid propellant rocket engine', noun-1, 'rocket engine', noun-1, []).
a_kind_of('spin canister', noun-1, can, noun-1, []).
a_kind_of('static firing', noun-1, launch, noun-1, []).
a_kind_of('surface vehicle', noun-1, vehicle, noun-1, []).
a_kind_of('surface-to-surface missile', noun-1, missile, noun-1, []).
a_kind_of('tail fin', noun-1, fin, noun-1, []).
a_kind_of('test firing', noun-1, launch, noun-1, []).
a_kind_of('test plan', noun-1, abst_obj, noun-1, []).
a_kind_of('test range', noun-1, range, noun-1, []).
a_kind_of('test vehicle', noun-1, vehicle, noun-1, []).
a_kind_of('time measuring instrument', noun-1, 'measuring instrument', noun-1, []).
a_kind_of('total view', noun-1, view, noun-1, []).
a_kind_of('tracking mount', noun-1, equipment, noun-1, []).
a_kind_of('training simulator', noun-1, simulator, noun-1, []).
a_kind_of('transport aircraft', noun-1, aircraft, noun-1, []).
a_kind_of('video screen', noun-1, screen, noun-1, []).
a_kind_of('video view', noun-1, view, noun-1, []).
a_kind_of('water vehicle', noun-1, vehicle, noun-1, []).
a_kind_of(account, infin-1, stative_act, infin-1, []).
a_kind_of(agree, infin-1, stative_act, infin-1, []).
a_kind_of(aid, infin-1, event_act, infin-1, []).
a_kind_of(air, noun-1, location, noun-1, []).
a_kind_of(aircraft, noun-1, phys_obj, noun-1, []).
a_kind_of(airport, noun-1, location, noun-1, []).
a_kind_of(angle, noun-1, abst_obj, noun-1, []).
a_kind_of(animal, noun-1, phys_obj, noun-1, []).
a_kind_of(antenna, noun-1, phys_obj, noun-1, []).
```

```
a_kind_of(appear, infin-1, stative_act, infin-1, []).
a_kind_of(approach, infin-1, ptrans, infin-1, []).
a_kind_of(arrive, infin-1, ptrans, infin-1, []).
a_kind_of(art, noun-1, abst_obj, noun-1, []).
a_kind_of(artist, noun-1, 'human being', noun-1, []).
a_kind_of(assemble, infin-1, pbuild, infin-1, [[assembly, noun-1]]).
a_kind_of(assembly, noun-1, event, noun-1, [[assemble, infin-1]]).
a_kind_of(assembly, noun-1, phys_obj, noun-1, [[assemble, infin-1]]).
a_kind_of(atrans, infin-1, event_act, infin-1, []).
a_kind_of(attack, infin-1, event_act, infin-1, [[attack, noun-1]]).
a_kind_of(attack, noun-1, event, noun-1, [[attack, infin-1]]).
a_kind_of(attempt, noun-1, event, noun-1, []).
a_kind_of(attend, infin-1, sensory_act, infin-1, []).
a_kind_of(attenuator, noun-1, phys_obj, noun-1, []).
a_kind_of(avoid, infin-1, event_act, infin-1, []).
a_kind_of(background, noun-1, position, noun-1, []).
a_kind_of(bank, noun-1, collection, noun-1, []).
a_kind_of(bay, noun-1, location, noun-1, []).
a_kind_of(be, infin-1, stative_act, infin-1, []).
a_kind_of(begin, infin-1, stative_act, infin-1, [[beginning, noun-1]]).
a_kind_of(beginning, noun-1, time, noun-1, [[begin, infin-1]]).
a_kind_of(blaze, noun-1, marking, noun-1, []).
a_kind_of(bloom, noun-1, event, noun-1, []).
a_kind_of(blow, infin-1, event_act, infin-1, []).
a_kind_of(board, noun-1, phys_obj, noun-1, []).
a_kind_of(body, noun-1, phys_obj, noun-1, []).
a_kind_of(bomb, noun-1, weapon, noun-1, []).
a_kind_of(bomb, noun-1, 'explosive device', noun-1, []).
a_kind_of(bottom, noun-1, position, noun-1, []).
a_kind_of(boxcar, noun-1, 'surface vehicle', noun-1, []).
a_kind_of(boy, noun-1, 'human being', noun-1, []).
a_kind_of(building, noun-1, phys_obj, noun-1, []).
a_kind_of(bulkhead, noun-1, phys_obj, noun-1, []).
a_kind_of(bureau_no, noun-1, number, noun-1, []).
a_kind_of(burn, noun-1, event, noun-1, [[heat, infin-1]]).
a_kind_of(camera, noun-1, 'photographic equipment', noun-1, []).
a_kind_of(can, noun-1, phys_obj, noun-1, []).
a_kind_of(canopy, noun-1, phys_obj, noun-1, []).
a_kind_of(carry, infin-1, ptrans, infin-1, []).
a_kind_of(cease, infin-1, stative_act, infin-1, []).
a_kind_of(center, noun-1, position, noun-1, []).
a_kind_of(change, infin-1, event_act, infin-1, []).
a_kind_of(checkerboard, noun-1, art, noun-1, []).
a_kind_of(chip, noun-1, phys_obj, noun-1, []).
a_kind_of(choose, infin-1, event_act, infin-1, []).
a_kind_of(circuit, noun-1, phys_obj, noun-1, []).
a_kind_of(clock, noun-1, 'time measuring instrument', noun-1, []).
a_kind_of(close, infin-1, stative_act, infin-1, []).
a_kind_of(clothing, noun-1, phys_obj, noun-1, []).
a_kind_of(cloud, noun-1, phys_obj, noun-1, []).
a_kind_of(cockpit, noun-1, phys_obj, noun-1, []).
a_kind_of(collection, noun-1, event, noun-1, []).
a_kind_of(color, infin-1, event_act, infin-1, []).
a_kind_of(color, noun-1, abst_obj, noun-1, []).
a_kind_of(communication_act, infin-1, event_act, infin-1, []).
a_kind_of(compare, infin-1, event_act, infin-1, [[comparison, noun-1]]).
a_kind_of(comparison, noun-1, event, noun-1, [[compare, infin-1]]).
a_kind_of(complete, infin-1, stative_act, infin-1, []).
a_kind_of(composite, noun-1, event, noun-1, []).
a_kind_of(compound, noun-1, phys_obj, noun-1, []).
a_kind_of(computer, noun-1, equipment, noun-1, []).
```

```
a_kind_of(conceal, infin-1, event_act, infin-1, []).
a_kind_of(conceive, infin-1, event_act, infin-1, [[conception, noun-1]]).
a_kind_of(conception, noun-1, event, noun-1, [[conceive, infin-1]]).
a_kind_of(configuration, noun-1, event, noun-1, []).
a_kind_of(connector, noun-1, phys_obj, noun-1, []).
a_kind_of(console, noun-1, equipment, noun-1, []).
a_kind_of(construction, noun-1, event, noun-1, [[pbuild, infin-1]]).
a_kind_of(continue, infin-1, stative_act, infin-1, []).
a_kind_of(coordinate, noun-1, position, noun-1, []).
a_kind_of(country, noun-1, location, noun-1, []).
a_kind_of(crane, noun-1, 'handling equipment', noun-1, []).
a_kind_of(crew, noun-1, 'human being', noun-1, []).
a_kind_of(curve, infin-1, event_act, infin-1, []).
a_kind_of(damage, noun-1, abst_obj, noun-1, []).
a_kind_of(damage, noun-1, event, noun-1,[[ break, infin-1]]).
a_kind_of(date, noun-1, position, noun-1, []).
a_kind_of(debris, noun-1, phys_obj, noun-1, []).
a_kind_of(deck, noun-1, phys_obj, noun-1, []).
a_kind_of(deck, noun-1, location, noun-1, []).
a_kind_of(decoy, noun-1, phys_obj, noun-1, []).
a_kind_of(decrease, infin-1, event_act, infin-1, [[erosion, noun-1]]).
a_kind_of(degree, noun-1, measure, noun-1, []).
a_kind_of(depart, infin-1, ptrans, infin-1, []).
a_kind_of(depend, infin-1, stative_act, infin-1, []).
a_kind_of(desert, noun-1, land, noun-1, []).
a_kind_of(destroyer, noun-1, ship, noun-1, []).
a_kind_of(detector, noun-1, phys_obj, noun-1, []).
a_kind_of(device, noun-1, phys_obj, noun-1, []).
a_kind_of(direct, infin-1, event_act, infin-1, []).
a_kind_of(disassemble, infin-1, pbuild, infin-1, [[disassembly, noun-1]]).
a_kind_of(disassembly, noun-1, event, noun-1, [[disassemble, infin-1]]).
a_kind_of(discover, infin-1, event_act, infin-1, []).
a_kind_of(dispenser, noun-1, phys_obj, noun-1, []).
a_kind_of(dispersion, noun-1, mixture, noun-1, []).
a_kind_of(divide, infin-1, event_act, infin-1, []).
a_kind_of(dolly, noun-1, 'surface vehicle', noun-1, []).
a_kind_of(dome, noun-1, phys_obj, noun-1, []).
a_kind_of(doubt, infin-1, stative_act, infin-1, []).
a_kind_of(drone, noun-1, phys_obj, noun-1, []).
a_kind_of(drop, noun-1, event, noun-1, [[launch, infin-1]]).
a_kind_of(dummy, noun-1, phys_obj, noun-1, []).
a_kind_of(edge, noun-1, position, noun-1, []).
a_kind_of(educate, infin-1, event_act, infin-1, [[education, noun-1]]).
a_kind_of(education, noun-1, event, noun-1, [[educate, infin-1]]).
a_kind_of(ejection, noun-1, event, noun-1, [[expel, infin-1], ['escape system', noun-
1]]).
a_kind_of(elevation, noun-1, location, noun-1, []).
a_kind_of(elevation, noun-1, event, noun-1, []).
a_kind_of(elevator, noun-1, phys_obj, noun-1, []).
a_kind_of(end, noun-1, position, noun-1, []).
a_kind_of(engine, noun-1, phys_obj, noun-1, []).
a_kind_of(equipment, noun-1, phys_obj, noun-1, []).
a_kind_of(erosion, noun-1, event, noun-1, [[erode, infin-1]]).
a_kind_of(evaluate, infin-1, event_act, infin-1, [[evaluation, noun-1]]).
a_kind_of(evaluation, noun-1, event, noun-1, [[evaluate, infin-1]]).
a_kind_of(evening, noun-1, time, noun-1, []).
a_kind_of(event, noun-1, abst_obj, noun-1, []).
a_kind_of(example, noun-1, kind, noun-1, []).
a_kind_of(exhaust, noun-1, smoke, noun-1, []).
a_kind_of(expand, infin-1, event_act, infin-1, []).
a_kind_of(expect, infin-1, stative_act, infin-1, []).
```

```
a_kind_of(expel, infin-1, event_act, infin-1, [[ejection, noun-1]]).
a_kind_of(explode, infin-1, stative_act, infin-1, [[explosion, noun-1]]).
a_kind_of(explosion, noun-1, event, noun-1, [[explode, infin-1]]).
a_kind_of(explosive, noun-1, 'chemical element', noun-1, []).
a_kind_of(farm, noun-1, location, noun-1, []).
a_kind_of(fastener, noun-1, phys_obj, noun-1, []).
a_kind_of(fatigue, infin-1, event_act, infin-1, []).
a_kind_of(feet, noun-1, measure, noun-1, []).
a_kind_of(fight, infin-1, event_act, infin-1, []).
a_kind_of(fin, noun-1, phys_obj, noun-1, []).
a_kind_of(fire, noun-1, phys_obj, noun-1, []).
a_kind_of(fireball, noun-1, phys_obj, noun-1, []).
a_kind_of(fix, infin-1, pbuild, infin-1, []).
a_kind_of(fleet, noun-1, organization, noun-1, []).
a_kind_of(flight, noun-1, event, noun-1, [[fly, infin-1]]).
a_kind_of(flower, noun-1, plant, noun-1, []).
a_kind_of(fly, infin-1, ptrans, infin-1, [[flight, noun-1]]).
a_kind_of(forget, infin-1, stative_act, infin-1, []).
a_kind_of(foreground, noun-1, position, noun-1, []).
a_kind_of(formation, noun-1, order, noun-1, []).
a_kind_of(fps, noun-1, measure, noun-1, []).
a_kind_of(frame, noun-1, phys_obj, noun-1, []).
a_kind_of(free, infin-1, stative_act, infin-1, []).
a_kind_of(front, noun-1, position, noun-1, []).
a_kind_of(fuel, noun-1, phys_obj, noun-1, []).
a_kind_of(generator, noun-1, equipment, noun-1, []).
a_kind_of(girl, noun-1, 'human being', noun-1, []).
a_kind_of(goggles, noun-1, clothing, noun-1, []).
a_kind_of(gulf, noun-1, location, noun-1, []).
a_kind_of(gun, noun-1, weapon, noun-1, []).
a_kind_of(hangar, noun-1, building, noun-1, []).
a_kind_of(haze, noun-1, cloud, noun-1, []).
a_kind_of(heat, infin-1, event_act, infin-1, [[burn, noun-1]]).
a_kind_of(heat, noun-1, abst_obj, noun-1, []).
a_kind_of(helicopter, noun-1, 'rotary wing aircraft', noun-1, []).
a_kind_of(heliostat, noun-1, instrument, noun-1, []).
a_kind_of(hit, infin-1, event_act, infin-1, []).
a_kind_of(hit, noun-1, event, noun-1, []).
a_kind_of(hoisting, noun-1, event, noun-1, []).
a_kind_of(horse, noun-1, mammal, noun-1, []).
a_kind_of(housing, noun-1, phys_obj, noun-1, []).
a_kind_of(id, noun-1, marking, noun-1, []).
a_kind_of(ignition, noun-1, event, noun-1, []).
a_kind_of(imitate, infin-1, event_act, infin-1, []).
a_kind_of(include, infin-1, stative_act, infin-1, []).
a_kind_of(increase, infin-1, event_act, infin-1, []).
a_kind_of(influence, infin-1, event_act, infin-1, []).
a_kind_of(ingest, infin-1, event_act, infin-1, [[ingestion, noun-1]]).
a_kind_of(ingestion, noun-1, event, noun-1, [[ingest, infin-1]]).
a_kind_of(inhabit, infin-1, stative_act, infin-1, []).
a_kind_of(injure, infin-1, event_act, infin-1, [[damage, noun-1]]).
a_kind_of(inquire, infin-1, communication_act, infin-1, []).
a_kind_of(inspect, infin-1, observe, infin-1, [[inspection, noun-1], [evaluation,
noun-1]]).
a_kind_of(inspection, noun-1, event, noun-1, [[inspect, infin-1]]).
a_kind_of(instrument, noun-1, phys_obj, noun-1, []).
a_kind_of(intensity, noun-1, measure, noun-1, []).
a_kind_of(inverter, noun-1, phys_obj, noun-1, []).
a_kind_of(keas, noun-1, measure, noun-1, []).
a_kind_of(kind, noun-1, abst_obj, noun-1, []).
a_kind_of(kts, noun-1, measure, noun-1, []).
```

```
a_kind_of(laboratory, noun-1, building, noun-1, []).
a_kind_of(lake, noun-1, location, noun-1, []).
a_kind_of(land, noun-1, location, noun-1, []).
a_kind_of(laser, noun-1, phys_obj, noun-1, []).
a_kind_of(launch, infin-1, event_act, infin-1, [[drop, noun-1], [launch, noun-1],
[shot, noun-1]]).
a_kind_of(launch, noun-1, event, noun-1, [[launch, infin-1]]).
a_kind_of(launcher, noun-1, phys_obj, noun-1, []).
a_kind_of(left, noun-1, position, noun-1, []).
a_kind_of(line, noun-1, marking, noun-1, []).
a_kind_of(loader, noun-1, 'surface vehicle', noun-1, []).
a_kind_of(luminaire, noun-1, 'lighting equipment', noun-1, []).
a_kind_of(mammal, noun-1, vertebrate, noun-1, []).
a_kind_of(marking, noun-1, abst_obj, noun-1, []).
a_kind_of(mbuild, infin-1, stative_act, infin-1, []).
a_kind_of(measure, infin-1, event_act, infin-1, []).
a_kind_of(measure, noun-1, abst_obj, noun-1, []).
a_kind_of(mercury_metal, noun-1, 'chemical element', noun-1, []).
a_kind_of(missile, noun-1, weapon, noun-1, []).
a_kind_of(mission, noun-1, project, noun-1, []).
a_kind_of(mixture, noun-1, event, noun-1, []).
a_kind_of(mm, noun-1, measure, noun-1, []).
a_kind_of(model, noun-1, phys_obj, noun-1, []).
a_kind_of(month, noun-1, time, noun-1, []).
%a_kind_of(motor, noun-1, phys_obj, noun-1, []).
a_kind_of(mountain, noun-1, location, noun-1, []).
a_kind_of(mtrans, infin-1, communication_act, infin-1, []).
a_kind_of(nose, noun-1, phys_obj, noun-1, []).
a_kind_of(number, infin-1, event_act, infin-1, [[number, noun-1]]).
a_kind_of(number, noun-1, marking, noun-1, [[number, infin-1]]).
a_kind_of(observe, infin-1, event_act, infin-1, []).
a_kind_of(ocean, noun-1, location, noun-1, []).
a_kind_of(occur, infin-1, stative_act, infin-1, []).
a_kind_of(open_act, infin-1, stative_act, infin-1, []).
a_kind_of(opening, noun-1, abst_obj, noun-1, []).
a_kind_of(operation, noun-1, event, noun-1, [[perform, infin-1]]).
a_kind_of(operator, noun-1, phys_obj, noun-1, []).
a_kind_of(order, noun-1, abst_obj, noun-1, []).
a_kind_of(ordnance, noun-1, weapon, noun-1, []).
a_kind_of(organization, noun-1, abst_obj, noun-1, []).
a_kind_of(overhead, noun-1, position, noun-1, []).
a_kind_of(overview, noun-1, view, noun-1, []).
a_kind_of(paint, noun-1, phys_obj, noun-1, []).
a_kind_of(pallet, noun-1, stand, noun-1, []).
a_kind_of(panel, noun-1, phys_obj, noun-1, []).
a_kind_of(parachute, noun-1, phys_obj, noun-1, []).
a_kind_of(path, noun-1, abst_obj, noun-1, []).
a_kind_of(pattern, noun-1, abst_obj, noun-1, []).
a_kind_of(payload, noun-1, phys_obj, noun-1, []).
a_kind_of(pbuild, infin-1, event_act, infin-1, [[construction, noun-1], [production,
noun-1]]).
a_kind_of(pedestal, noun-1, stand, noun-1, []).
a_kind_of(people, noun-1, 'human being', noun-1, []).
a_kind_of(perform, infin-1, event_act, infin-1, [[operation, noun-1]]).
a_kind_of(permit, infin-1, stative_act, infin-1, []).
a_kind_of(photograph, noun-1, event, noun-1, []).
a_kind_of(pilot, noun-1, 'human being', noun-1, []).
a_kind_of(plant, noun-1, phys_obj, noun-1, []).
a_kind_of(plaque, noun-1, phys_obj, noun-1, []).
a_kind_of(plastisol, noun-1, dispersion, noun-1, []).
a_kind_of(plume, noun-1, phys_obj, noun-1, []).
```

```
a_kind_of(pod, noun-1, 'external store', noun-1, []).
a_kind_of(pole, noun-1, phys_obj, noun-1, []).
a_kind_of(position, noun-1, location, noun-1, []).
a_kind_of(power, noun-1, abst_obj, noun-1, []).
a_kind_of(practice, noun-1, abst_obj, noun-1, []).
a_kind_of(precipitation, noun-1, phys_obj, noun-1, []).
a_kind_of(prepare, infin-1, event_act, infin-1, []).
a_kind_of(primate, noun-1, mammal, noun-1, []).
a_kind_of(production, noun-1, event, noun-1, [[pbuild, infin-1]]).
a_kind_of(program, noun-1, abst_obj, noun-1, []).
a_kind_of(progress, noun-1, abst_obj, noun-1, []).
a_kind_of(project, noun-1, abst_obj, noun-1, []).
a_kind_of(projectile, noun-1, phys_obj, noun-1, []).
a_kind_of(propel, infin-1, event_act, infin-1, []).
a_kind_of(ptrans, infin-1, event_act, infin-1, []).
a_kind_of(pursue, infin-1, event_act, infin-1, []).
a_kind_of(pylon, noun-1, phys_obj, noun-1, []).
a_kind_of(rack, noun-1, stand, noun-1, []).
a_kind_of(radar, noun-1, phys_obj, noun-1, []).
a_kind_of(radome, noun-1, housing, noun-1, []).
a_kind_of(range, noun-1, location, noun-1, []).
a_kind_of(rank, noun-1, order, noun-1, []).
a_kind_of(rear, noun-1, position, noun-1, []).
a_kind_of(red, noun-1, color, noun-1, []).
a_kind_of(relation, noun-1, abst_obj, noun-1, []).
a_kind_of(release, infin-1, event_act, infin-1, []).
a_kind_of(remains, noun-1, phys_obj, noun-1, []).
a_kind_of(request, infin-1, communication_act, infin-1, []).
a_kind_of(require, infin-1, stative_act, infin-1, []).
a_kind_of(resistor, noun-1, attenuator, noun-1, []).
a_kind_of(revetment, noun-1, phys_obj, noun-1, []).
a_kind_of(ride, infin-1, ptrans, infin-1, []).
a_kind_of(right, noun-1, position, noun-1, []).
a_kind_of(river, noun-1, location, noun-1, []).
a_kind_of(rock, noun-1, phys_obj, noun-1, []).
a_kind_of(rocket, noun-1, phys_obj, noun-1, []).
a_kind_of(room, noun-1, phys_obj, noun-1, []).
a_kind_of(round, noun-1, event, noun-1, [[fire, infin-1]]).
a_kind_of(run, noun-1, event, noun-1, [[ptrans, infin-1]]).
a_kind_of(runway, noun-1, location, noun-1, []).
a_kind_of(screen, noun-1, phys_obj, noun-1, []).
a_kind_of(sea, noun-1, location, noun-1, []).
a_kind_of(seat, noun-1, phys_obj, noun-1, []).
a_kind_of(sensory_act, infin-1, event_act, infin-1, []).
a_kind_of(sequence, noun-1, order, noun-1, []).
a_kind_of(serial_no, noun-1, number, noun-1, []).
a_kind_of(setup, noun-1, abst_obj, noun-1, []).
a_kind_of(ship, noun-1, 'water vehicle', noun-1, []).
a_kind_of(ship, noun-1, phys_obj, noun-1, []).
a_kind_of(shock, infin-1, communication_act, infin-1, []).
a_kind_of(shot, noun-1, view, noun-1, [[launch, infin-1]]).
a_kind_of(show, infin-1, communication_act, infin-1, []).
a_kind_of(side, noun-1, position, noun-1, []).
a_kind_of(silver, noun-1, color, noun-1, []).
a_kind_of(simulator, noun-1, phys_obj, noun-1, []).
a_kind_of(skid, noun-1, phys_obj, noun-1, []).
a_kind_of(sled, noun-1, 'surface vehicle', noun-1, []).
a_kind_of(smoke, noun-1, plastisol, noun-1, []).
a_kind_of(smoulder, noun-1, event, noun-1, []).
a_kind_of(snow, noun-1, precipitation, noun-1, []).
a_kind_of(solder, infin-1, pbuild, infin-1, [[solder, noun-1], [soldering, noun-1]]).
```

```
a_kind_of(solder, noun-1, phys_obj, noun-1, [[solder, infin-1]]).
a_kind_of(soldering, noun-1, event, noun-1, [[solder, infin-1]]).
a_kind_of(sort, noun-1, kind, noun-1, []).
a_kind_of(sound, noun-1, location, noun-1, []).
a_kind_of(spread, infin-1, event_act, infin-1, [[spread, noun-1], [spreading, noun-
1]]).
a_kind_of(spread, noun-1, event, noun-1, [[spread, infin-1]]).
a_kind_of(speak, infin-1, communication_act, infin-1, []).
a_kind_of(spreading, noun-1, event, noun-1, [[spread, infin-1]]).
a_kind_of(squadron, noun-1, organization, noun-1, []).
a_kind_of(stand, noun-1, phys_obj, noun-1, []).
a_kind_of(station, noun-1, phys_obj, noun-1, []).
a_kind_of(store, infin-1, event_act, infin-1, []).
a_kind_of(strobe, noun-1, 'optical equipment', noun-1, []).
a_kind_of(study, noun-1, project, noun-1, []).
a_kind_of(submarine, noun-1, ship, noun-1, []).
a_kind_of(sunset, noun-1, time, noun-1, []).
a_kind_of(survive, infin-1, stative_act, infin-1, [[survivability, noun-1]]).
a_kind_of(survivability, noun-1, event, noun-1, [[survive, infin-1]]).
a_kind_of(switch, infin-1, ptrans, infin-1, []).
a_kind_of(system, noun-1, phys_obj, noun-1, []).
a_kind_of(tail, noun-1, phys_obj, noun-1, []).
a_kind_of(tail, noun-1, position, noun-1, []).
a_kind_of(takeoff, noun-1, event, noun-1, []).
a_kind_of(tank, noun-1, 'combat vehicle', noun-1, []).
a_kind_of(target, noun-1, phys_obj, noun-1, []).
%a_kind_of(test, infin-1, event_act, infin-1, [[test, noun-1]]).
a_kind_of(test, noun-1, event, noun-1, [[test, infin-1]]).
a_kind_of(tilt, noun-1, angle, noun-1, []).
a_kind_of(time, noun-1, abst_obj, noun-1, []).
a_kind_of(num_stamp, noun-1, number, noun-1, []).
a_kind_of(top, noun-1, position, noun-1, []).
a_kind_of(tower, noun-1, phys_obj, noun-1, []).
a_kind_of(track, noun-1, phys_obj, noun-1, []).
a_kind_of(trail, noun-1, path, noun-1, []).
a_kind_of(trailer, noun-1, 'motor vehicle', noun-1, []).
a_kind_of(train, noun-1, 'motor vehicle', noun-1, []).
a_kind_of(training, noun-1, event, noun-1, []).
a_kind_of(tree, noun-1, plant, noun-1, []).
a_kind_of(truck, noun-1, 'motor vehicle', noun-1, []).
a_kind_of(try, infin-1, event_act, infin-1, []).
a_kind_of(underside, noun-1, position, noun-1, []).
a_kind_of(use, infin-1, stative_act, infin-1, []).
a_kind_of(validate, infin-1, event_act, infin-1, [[validation, noun-1]]).
a_kind_of(validation, noun-1, event, noun-1, [[validate, noun-1]]).
a_kind_of(valley, noun-1, land, noun-1, []).
a_kind_of(vehicle, noun-1, phys_obj, noun-1, []).
a_kind_of(vertebrate, noun-1, animal, noun-1, []).
a_kind_of(view, noun-1, event, noun-1, []).
a_kind_of(vision, noun-1, abst_obj, noun-1, []).
a_kind_of(wait, infin-1, stative_act, infin-1, []).
a_kind_of(warhead, noun-1, phys_obj, noun-1, []).
a_kind_of(wash, noun-1, land, noun-1, []).
a_kind_of(water, noun-1, phys_obj, noun-1, []).
a_kind_of(weapon, noun-1, phys_obj, noun-1, []).
a_kind_of(wear, infin-1, event_act, infin-1, []).
a_kind_of(wildflower, noun-1, flower, noun-1, []).
a_kind_of(wing, noun-1, phys_obj, noun-1, []).
a_kind_of(wingtip, noun-1, phys_obj, noun-1, []).
a_kind_of(wire, noun-1, phys_obj, noun-1, []).
a_kind_of(worker, noun-1, 'human being', noun-1, []).
```

```
a_kind_of(year, noun-1, time, noun-1, []).

%%% ***************  SLOT Definitions  ***************

%%% Slot Fields Defined.

%%% slot(Word, WordKey, transforms,
%%%      [c(OldCase, OldReln, OldClass, OldKey,
%%%         AssocKase, AssocReln, AssocClass, AssocKey,
%%%         NewKase, NewReln, NewClass, NewKey)]).

slot(e_r_concept, root-1, correlations,
    []).
slot(e_r_concept, root-1, inner_cases,
    []).
slot(e_r_concept, root-1, case_vals,
    []).
slot(obj_concept, noun-1, inner_cases,
    [c(after, event, time),
     c(before, event, time)]).
slot(abst_obj, noun-1, inner_cases,
    [c(behind, _, location),
     c(in, _, location),
     c(on, _, location),
     c(over, _, location),
     c(under, _, location)]).
slot(position, noun-1, correlations,
    [c(part_of, order)]).
slot(order, noun-1, inner_cases,
    [c(from, _, source)]).
slot(order, noun-1, correlations,
    [c(has_part, position)]).
slot('Armitage Field', noun-1, correlations,
    [c(has_part, 'Cold Line'),
     c(has_part, 'Hot Line')]).
slot('Cold Line', noun-1, correlations,
    [c(part_of, 'Armitage Field')]).
slot('Hot Line', noun-1, correlations,
    [c(part_of, 'Armitage Field')]).
slot('USMC', noun-1, correlations,
    [c(has_part, 'VMA-513')]).
slot('USN', noun-1, correlations,
    [c(has_part, 'NWC'),
     c(has_part, 'Seventh Fleet')]).
slot('Seventh Fleet', noun-1, correlations,
    [c(part_of, 'USN')]).
slot('Skyray', noun-1, correlations,
    [c(has_part, 'fiber optic')]).
slot('VMA-513', noun-1, correlations,
    [c(part_of, 'USMC')]).
slot('VX-5', noun-1, correlations,
    [c(part_of, 'NWC')]).
slot('RAPEC', noun-1, correlations,
    [c(program_about, ejection)]).
slot('fiber optic', noun-1, correlations,
    [c(part_of, 'Skyray')]).
slot(engine, noun-1, correlations,
    [c(consumes, fuel)]).
slot(event, noun-1, inner_cases,
    [c(against, _, destination),
     c(at, phys_obj, destination),
```

```
     c(at, location, location),
     c(by, 'human being', agent),
     c(by, phys_obj, force),
     c(during, _, time),
     c(from, _, source),
     c(in, num, time),
     c(of, event, theme),
     c(of, location, theme),
     c(of, marking, theme),
     c(of, order, theme),
     c(of, path, theme),
     c(of, phys_obj, theme),
     c(on, date, time),
     c(on, phys_obj, location),
     c(to, phys_obj, theme),
     c(with, _, accompaniment)]).
slot('air-to-air view', noun-1, infer,
     [c(theme, _, _, _,
        location, in, air, noun-1)]).
slot(ejection, noun-1, correlations,
     [c(related_program, 'RAPEC')]).
slot(fuel, noun-1, correlations,
     [c(consumed_by, engine)]).
slot(panel, noun-1, correlations,
     [c(has_part, 'photovoltaic cell')]).
slot('photovoltaic cell', noun-1, correlations,
     [c(part_of, panel)]).
slot(relation, noun-1, inner_cases,
     [c(of, _, theme)]).
slot(location, noun-1, inner_cases,
     [c(from, _, location),
      c(in, _, location),
      c(under, _, location)]).
slot(airport, noun-1, correlations,
     [c(has_part, runway),
      c(has_part, tower)]).
slot('NWC', noun-1, correlations,
     [c(has_part, 'Coso Range'),
      c(has_part, 'G Range'),
      c(has_part, 'VX-5')]).
slot('Coso Range', noun-1, correlations,
     [c(part_of, 'NWC')]).
slot('G Range', noun-1, correlations,
     [c(has_part, 'G-1 Range'),
      c(part_of, 'NWC')]).
slot('G-1 Range', noun-1, correlations,
     [c(has_part, 'T-5'),
      c(part_of, 'G Range')]).
slot(runway, noun-1, correlations,
     [c(part_of, airport)]).
slot(phys_obj, noun-1, inner_cases,
     [c(against, phys_obj, destination),
      c(after, time, time),
      c(at, event, time),
      c(at, measure, attribute),
      c(at, _, location),
      c(before, phys_obj, location),
      c(for, event, reason),
      c(from, _, source),
      c(in, _, location),
      c(behind, _, location),
```

```
      c(on, date, time),
      c(on, _, location),
      c(outside, _, location),
      c(over, location, location),
      c(over, phys_obj, location),
      c(to, phys_obj, destination),
      c(under, _, location)]).
slot(aircraft, noun-1, correlations,
     [c(has_part, 'centerline station'),
      c(has_part, body),
      c(has_part, cockpit),
      c(has_part, 'escape system'),
      c(has_part, instrument),
      c(has_part, frame),
      c(has_part, nose),
      c(has_part, seat),
      c(has_part, side),
      c(has_part, tail),
      c(has_part, wing)]).
slot(building, noun-1, correlations,
     [c(has_part, room)]).
slot(room, noun-1, correlations,
     [c(part_of, building)]).
slot('rotary wing aircraft', noun-1, correlations,
     [c(has_part, 'rotary wing')]).
slot(animal, noun-1, correlations,
     [c(has_part, body),
      c(has_part, nose)]).
slot(crew, noun-1, case_vals,
     [quantity(plural)]).
slot(body, noun-1, correlations,
     [c(part_of, animal),
      c(part_of, vehicle)]).
slot(bulkhead, noun-1, correlations,
     [c(part_of, bomb),
      c(part_of, missile),
      c(part_of, ship)]).
slot(canopy, noun-1, correlations,
     [c(part_of, 'attack aircraft')]).
slot(cockpit, noun-1, correlations,
     [c(part_of, aircraft)]).
slot(deck, noun-1, correlations,
     [c(part_of, ship)]).
slot(dome, noun-1, correlations,
     [c(part_of, missile)]).
slot(fire, noun-1, inner_cases,
     [c(from, spreading, destination)]).
slot(clothing, noun-1, correlations,
     [c(worn_by, 'human being')]).
slot('flight deck', noun-1, correlations,
     [c(part_of, 'aircraft carrier')]).
slot('hangar bay', noun-1, correlations,
     [c(part_of, 'aircraft carrier')]).
slot('human being', noun-1, correlations,
     [c(wears, clothing)]).
slot('escape system', noun-1, correlations,
     [c(part_of, aircraft)]).
slot('attack aircraft', noun-1, correlations,
     [c(has_part, pod),
      c(has_part, missile),
      c(has_part, canopy)]).
```

```
slot(pod, noun-1, correlations,
     [c(part_of, 'attack aircraft'),
      c(has_part, 'FLIR detector')]).
slot(pod, noun-1, transforms,
     [c(attribute, part_of, aircraft, noun-1,
        _, _, _, _,
        location, on, _, _)]).
slot(frame, noun-1, correlations,
     [c(part_of, aircraft)]).
slot(laser, noun-1, correlations,
     [c(part_of, 'Maverick')]).
slot(side, noun-1, correlations,
     [c(part_of, aircraft)]).
slot('Dev-Assist', noun-1, correlations,
     [c(part_of, missile)]).
slot('FLIR detector', noun-1, correlations,
     [c(part_of, pod)]).
slot('guided missile submarine', noun-1, correlations,
     [c(has_part, 'FBM')]).
slot(engine, noun-1, correlations,
     [c(part_of, missile)]).
slot(missile, noun-1, correlations,
     [c(has_part, bulkhead),
      c(has_part, 'Dev-Assist'),
      c(has_part, dome),
      c(has_part, engine),
      c(has_part, 'homing device'),
      c(has_part, 'tail fin'),
      c(has_part, warhead),
      c(has_part, 'TDD'),
      c(part_of, 'attack aircraft')]).
slot(missile, noun-1, transforms,
     [c(attribute, part_of, aircraft, noun-1,
        _, _, _, _,
        location, on, _, _)]).
slot(missile, noun-1, infer,
     [c(activity, _, assemble, infin-1,
        location, on, aircraft, noun-1)]).
slot(bomb, noun-1, infer,
     [c(activity, _, assemble, infin-1,
        location, on, aircraft, noun-1)]).
slot(pod, noun-1, infer,
     [c(activity, _, assemble, infin-1,
        location, on, aircraft, noun-1)]).
slot(seat, noun-1, transforms,
     [c(attribute, part_of, aircraft, noun-1,
        _, _, _, _,
        location, on, _, _)]).
slot(weapon, noun-1, transforms,
     [c(attribute, part_of, aircraft, noun-1,
        _, _, _, _,
        location, on, _, _)]).
slot(tower, noun-1, transforms,
     [c(attribute, part_of, building, noun-1,
        _, _, _, _,
        location, at, _, _)]).
slot('FBM', noun-1, correlations,
     [c(part_of, 'guided missile submarine')]).
slot('homing device', noun-1, correlations,
     [c(part_of, missile)]).
slot(nose, noun-1, correlations,
```

```
    [c(part_of, aircraft),
     c(part_of, animal)]).
slot(tail, noun-1, correlations,
    [c(part_of, aircraft)]).
slot(launcher, noun-1, correlations,
    [c(part_of, wing)]).
slot(pedestal, noun-1, correlations,
    [c(part_of, radar)]).
slot(pylon, noun-1, correlations,
    [c(part_of, wing)]).
slot(radar, noun-1, correlations,
    [c(has_part, pedestal),
     c(part_of, truck)]).
slot(ship, noun-1, correlations,
    [c(has_part, bulkhead),
     c(has_part, deck)]).
slot('centerline station', noun-1, correlations,
    [c(part_of, aircraft)]).
slot(tower, noun-1, correlations,
    [c(part_of, airport),
     c(has_part, antenna)]).
slot(truck, noun-1, correlations,
    [c(has_part, radar)]).
slot(antenna, noun-1, correlations,
    [c(part_of, tower)]).
slot('Bigeye', noun-1, correlations,
    [c(has_part, 'chemical element')]).
slot('TDD', noun-1, correlations,
    [c(part_of, missile)]).
slot('chemical element', noun-1, correlations,
    [c(part_of, 'Bigeye')]).
slot(warhead, noun-1, correlations,
    [c(part_of, missile)]).
slot(bomb, noun-1, correlations,
    [c(has_part, bulkhead),
     c(has_part, explosive),
     c(has_part, opening),
     c(has_part, 'tail fin'),
     c(held_by, rack)]).
slot(explosive, noun-1, correlations,
    [c(part_of, bomb)]).
slot(instrument, noun-1, correlations,
    [c(part_of, aircraft)]).
slot(opening, noun-1, correlations,
    [c(part_of, bomb)]).
slot(rack, noun-1, correlations,
    [c(holds, bomb)]).
slot(wing, noun-1, correlations,
    [c(has_part, pylon),
     c(has_part, launcher),
     c(has_part, wingtip),
     c(part_of, aircraft)]).
slot(wingtip, noun-1, correlations,
    [c(part_of, wing)]).
slot('tail fin', noun-1, correlations,
    [c(part_of, bomb),
     c(part_of, missile)]).
slot('rotary wing', noun-1, correlations,
    [c(part_of, 'rotary wing aircraft'),
     c(has_part, 'rotary wing dispenser')]).
slot('rotary wing dispenser', noun-1, correlations,
```

```
     [c(part_of, 'rotary wing'),
      c(has_part, 'spin canister')]).
slot('spin canister', noun-1, correlations,
     [c(part_of, 'rotary wing dispenser')]).
slot(train, noun-1, correlations,
     [c(has_part, boxcar)]).
slot(boxcar, noun-1, correlations,
     [c(part_of, train)]).
slot('aircraft carrier', noun-1, correlations,
     [c(has_part, 'flight deck'),
      c(has_part, 'hangar bay')]).
slot('Maverick', noun-1, correlations,
     [c(has_part, laser),
      c(has_part, 'infrared detector')]).
slot('T-5', noun-1, correlations,
     [c(part_of, 'G-1 Range')]).
slot(seat, noun-1, correlations,
     [c(has_part, rocket),
      c(part_of, aircraft)]).
slot(rocket, noun-1, correlations,
     [c(part_of, seat)]).
slot(act_concept, infin-1, inner_cases,
     [c(subj, _, agent),
      c(obj, _, object),
      c(after, event, time),
      c(before, event, time),
      c(before, phys_obj, location),
      c(behind, _, location),
      c(from, phys_obj, source),
      c(at, time, time),
      c(on, time, time),
      c(over, phys_obj, location),
      c(under, phys_obj, location),
      c(bottom_up, _, manner),
      c(front_to_back, _, manner),
      c(inside_out, _, manner),
      c(left_to_right, _, manner),
      c(right_to_left, _, manner),
      c(top_down, _, manner),
      c(top_to_bottom, _, manner)]).
slot(event_act, infin-1, inner_cases,
     [c(against, _, destination),
      c(for, event, reason),
      c(for, _, beneficiary),
      c(with, phys_obj, instrument),
      c(at, location, location),
      c(at, phys_obj, location),
      c(by, 'human being', agent),
      c(in, _, location),
      c(on, _, location),
      c(obj, _, theme),
      c(over, location, location),
      c(to, phys_obj, destination),
      c(to, location, destination)]).
slot(launch, infin-1, transforms,
     [c(agent, _, aircraft, noun-1,
        theme, _, missile, noun-1,
        source, from, _, _)]).
slot(launch, noun-1, inner_cases,
     [c(on, phys_obj, destination)]).
slot(launch, infin-1, inner_cases,
```

```
    [c(at, phys_obj, destination)]).
slot(flight, noun-1, transforms,
    [c(theme, _, aircraft, noun-1,
       _, _, _, _,
       agent, _, aircraft, noun-1),
     c(theme, _, missile, noun-1,
       _, _, _, _,
       agent, _, missile, noun-1),
     c(inst, _, flight, noun-1,
       _, _, _, _,
       activity, _, fly, infin-1)]).
slot(fly, infin-1, inner_cases,
    [c(with, path, accompaniment)]).
slot(launch, noun-1, transforms,
    [c(inst, _, launch, noun-1,
       _, _, _, _,
       activity, _, launch, infin-1)]).
slot(hit, noun-1, transforms,
    [c(inst, _, hit, noun-1,
       _, _, _, _,
       activity, _, hit, infin-1)]).
slot(soldering, noun-1, transforms,
    [c(inst, _, soldering, noun-1,
       agent, _, 'human being', noun-1,
       activity, _, solder, infin-1)]).
slot('test firing', noun-1, transforms,
    [c(inst, _, 'test firing', noun-1,
       _, _, _, _,
       activity, _, launch, infin-1)]).
slot(stative_act, infin-1, inner_cases,
    [c(above, phys_obj, location),
     c(from, event, destination),
     c(obj, event, theme),
     c(obj, location, theme),
     c(obj, phys_obj, theme),
     c(on, location, location)]).
slot(open_act, infin-1, inner_cases,
    [c(on, phys_obj, location)]).
slot(communication_act, infin-1, inner_cases,
    [c(in, _, language),
     c(on, _, topic),
     c(to, _, recipient)]).
slot(ptrans, infin-1, inner_cases,
    [c(from, location, source),
     c(obj, location, source),
     c(obj, phys_obj, theme),
     c(subj, phys_obj, agent),
     c(to, location, destination),
     c(toward, phys_obj, destination)]).
slot(depart, infin-1, inner_cases,
    [c(obj, location, source),
     c(obj, phys_obj, source)]).
slot(fly, infin-1, inner_cases,
    [c(with, trail, accompaniment)]).
slot(carry, infin-1, inner_cases,
    [c(obj, phys_obj, object)]).
slot(carry, infin-1, transforms,
    [c(agent, _, aircraft, noun-1,
       theme, _, missile, noun-1,
       location, on, assemble, infin-1)]).
slot(pbuild, infin-1, inner_cases,
```

```
     [c(from, _, source),
      c(with, phys_obj, object),
      c(obj, phys_obj, theme),
      c(to, phys_obj, destination)]).
slot(assemble, infin-1, transforms,
     [c(theme, _, aircraft, noun-1,
        object, with, missile, noun-1,
        location, on, _, _)]).
slot(observe, infin-1, inner_cases,
     [c(obj, phys_obj, theme),
      c(from, position, source)]).
slot(approach, infin-1, inner_cases,
     [c(obj, phys_obj, destination)]).
```

APPENDIX F

LOGICAL FORMS

```
::::::::::::::
10851
::::::::::::::

lf(inst('noun(10851-1-7)','CVA-16')).
lf(location('noun(10851-1-2)',on('noun(10851-1-7)'))).
lf(location('noun(10851-1-2)',outside('noun(10851-1-5)'))).
lf(location('noun(10851-1-2)',in('noun(10851-1-4)'))).
lf(location('noun(10851-1-2)',on('noun(10851-1-3)'))).
lf(inst('noun(10851-1-2)','Bullpup')).
lf(inst('noun(10851-1-4)','hangar bay')).
lf(inst('noun(10851-1-3)','bomb skid')).
lf(inst('noun(10851-1-5)',elevator)).
::::::::::::::
10862
::::::::::::::

lf(attribute('noun(10862-2-5)',inst('noun(10862-2-6)'))).
lf(attribute('noun(10862-2-5)',inst('noun(10862-2-4)'))).
lf(inst('noun(10862-2-5)','F-3H-1')).
lf(theme('noun(10862-3-1)',obj('noun(10862-1-1)'))).
lf(time('noun(10862-3-1)',during('noun(10862-3-3)'))).
lf(source('noun(10862-3-1)',from('noun(10862-3-2)'))).
lf(inst('noun(10862-3-1)','air-to-air view')).
lf(attribute('noun(10862-2-4)','137010')).
lf(inst('noun(10862-2-4)',bureau_no)).
lf(inst('noun(10862-1-3)','Seventh Fleet')).
lf(quantity('noun(10862-1-1)',plural(4))).
lf(location('noun(10862-1-1)',on('noun(10862-2-2)'))).
lf(quantity('noun(10862-1-1)',1)).
lf(location('noun(10862-1-1)',in('noun(10862-3-a3)'))).
lf(inst('noun(10862-1-1)','Sparrow 3')).
lf(inst('noun(10862-3-2)',side)).
lf(attribute('noun(10862-2-6)','7010')).
lf(location('noun(10862-2-6)',on('noun(10862-2-7)'))).
lf(inst('noun(10862-2-6)','Point Mugu')).
lf(location('noun(10862-2-2)',on('noun(10862-2-5)'))).
lf(inst('noun(10862-2-2)',underside)).
lf(attribute('noun(10862-1-4)',western)).
lf(inst('noun(10862-1-4)','Pacific Ocean')).
lf(inst('noun(10862-3-a3)',air)).
lf(theme('noun(10862-3-3)',obj('noun(10862-1-1)'))).
lf(activity('noun(10862-3-3)',launch)).
lf(inst('noun(10862-2-7)',tail)).
lf(agent('adj(10862-1-1)',obj('noun(10862-1-3)'))).
lf(agent('adj(10862-1-1)',obj('noun(10862-1-1)'))).
lf(location('adj(10862-1-1)',in('noun(10862-1-4)'))).
lf(activity('adj(10862-1-1)',perform)).
::::::::::::::
10880
::::::::::::::

lf(inst('noun(10880-2-a5)',artist)).
```

```
lf(attribute('noun(10880-2-5)',isa('noun(10880-2-6)'))).
lf(inst('noun(10880-2-5)','SSG-N1')).
lf(attribute('noun(10880-2-a4)',part_of('noun(10880-2-5)'))).
lf(inst('noun(10880-2-a4)','FBM')).
lf(agent('noun(10880-2-2)',obj('noun(10880-2-a5)'))).
lf(theme('noun(10880-2-2)',obj('noun(10880-2-5)'))).
lf(inst('noun(10880-2-2)',conception)).
lf(agent('prespart(10880-3-1)',obj('noun(10880-2-5)'))).
lf(theme('prespart(10880-3-1)',obj('noun(10880-2-a4)'))).
lf(activity('prespart(10880-3-1)',launch)).
lf(attribute('noun(10880-2-6)',nuclear)).
lf(inst('noun(10880-2-6)',program)).
lf(attribute('noun(10880-1-1)',graphics)).
lf(inst('noun(10880-1-1)',art)).
::::::::::::::
110169
::::::::::::::

lf(quantity('noun(110169-2-1)',closeup)).
lf(attribute('noun(110169-2-1)',excellent)).
lf(theme('noun(110169-2-1)',obj('noun(110169-1-1)'))).
lf(inst('noun(110169-2-1)','side view')).
lf(attribute('noun(110169-1-6)',has_part('noun(110169-1-7)'))).
lf(attribute('noun(110169-1-6)',inst('noun(110169-1-8)'))).
lf(attribute('noun(110169-1-6)',inst('noun(110169-1-5)'))).
lf(inst('noun(110169-1-6)','A-6A')).
lf(location('noun(110169-1-8)',on('noun(110169-1-9)'))).
lf(inst('noun(110169-1-8)','NG')).
lf(attribute('noun(110169-1-5)','151562')).
lf(inst('noun(110169-1-5)',bureau_no)).
lf(theme('pastpart(110169-3-1)',obj('noun(110169-1-3)'))).
lf(location('pastpart(110169-3-1)',on('noun(110169-3-2)'))).
lf(location('pastpart(110169-3-1)',on('noun(110169-1-6)'))).
lf(activity('pastpart(110169-3-1)',assemble)).
lf(quantity('noun(110169-1-3)',plural)).
lf(location('noun(110169-1-3)',on('noun(110169-1-6)'))).
lf(inst('noun(110169-1-3)','Standard Arm')).
lf(quantity('noun(110169-1-1)',plural)).
lf(inst('noun(110169-1-1)','Bat')).
lf(theme('prespart(110169-4-1)',obj('noun(110169-1-1)'))).
lf(location('prespart(110169-4-1)',on('noun(110169-4-3)'))).
lf(activity('prespart(110169-4-1)',inhabit)).
lf(location('noun(110169-4-3)',under('noun(110169-3-2)'))).
lf(inst('noun(110169-4-3)',land)).
lf(inst('noun(110169-3-2)',wing)).
lf(inst('noun(110169-1-9)',tail)).
lf(attribute('noun(110169-1-7)','562')).
lf(inst('noun(110169-1-7)',nose)).
::::::::::::::
124
::::::::::::::

lf(inst('noun(124-3-1)','human being')).
lf(theme('noun(124-2-1)',obj('noun(124-2-2)'))).
lf(inst('noun(124-2-1)',view)).
lf(inst('noun(124-1-2)','Harvey Field')).
lf(agent('prespart(124-3-1)',obj('noun(124-3-1)'))).
lf(activity('prespart(124-3-1)',perform)).
lf(attribute('noun(124-2-4)',inside)).
lf(inst('noun(124-2-4)',tower)).
lf(source('noun(124-2-3)',from('noun(124-2-4)'))).
```

```
lf(attribute('noun(124-2-3)',part_of('noun(124-2-2)'))).
lf(inst('noun(124-2-3)',aircraft)).
lf(inst('noun(124-2-2)',runway)).
lf(quantity('noun(124-1-1)',plural)).
lf(location('noun(124-1-1)',at('noun(124-1-2)'))).
lf(inst('noun(124-1-1)',equipment)).
::::::::::::::
161044
::::::::::::::

lf(attribute('noun(161044-1-5)','BQM control')).
lf(attribute('noun(161044-1-5)',owned_by('noun(161044-1-a21)'))).
lf(attribute('noun(161044-1-5)',inst('noun(161044-1-3)'))).
lf(inst('noun(161044-1-5)','UH-2A')).
lf(theme('noun(161044-1-6)',obj('noun(161044-1-5)'))).
lf(quantity('noun(161044-1-6)',full)).
lf(inst('noun(161044-1-6)','side view')).
lf(attribute('noun(161044-1-3)','149033')).
lf(inst('noun(161044-1-3)',bureau_no)).
lf(inst('noun(161044-1-a21)','USN')).
::::::::::::::
161045
::::::::::::::

lf(attribute('noun(161045-1-6)',owned_by('noun(161045-1-a22)'))).
lf(attribute('noun(161045-1-6)',inst('noun(161045-1-4)'))).
lf(inst('noun(161045-1-6)','C-130A')).
lf(inst('noun(161045-1-a22)','VC-3')).
lf(attribute('noun(161045-1-4)','158228')).
lf(inst('noun(161045-1-4)',bureau_no)).
lf(quantity('noun(161045-2-1)',closeup)).
lf(theme('noun(161045-2-1)',obj('noun(161045-1-2)'))).
lf(theme('noun(161045-2-1)',obj('noun(161045-2-4)'))).
lf(inst('noun(161045-2-1)',view)).
lf(theme('pastpart(161045-1-1)',obj('noun(161045-1-2)'))).
lf(location('pastpart(161045-1-1)',on('noun(161045-1-6)'))).
lf(activity('pastpart(161045-1-1)',assemble)).
lf(inst('noun(161045-1-2)','BQM')).
lf(attribute('noun(161045-2-4)',part_of('noun(161045-1-6)'))).
lf(inst('noun(161045-2-4)',wing)).
::::::::::::::
161082
::::::::::::::

lf(inst('noun(161082-1-7)','HH-1K')).
lf(inst('noun(161082-2-a23)',tank)).
lf(inst('noun(161082-1-4)','AIM-9L')).
lf(theme('noun(161082-1-1)',obj('noun(161082-1-5)'))).
lf(inst('noun(161082-1-1)','air-to-air view')).
lf(agent('prespart(161082-2-1)',obj('noun(161082-1-4)'))).
lf(destination('prespart(161082-2-1)',to('noun(161082-2-3)'))).
lf(activity('prespart(161082-2-1)',arrive)).
lf(inst('noun(161082-1-a24)',air)).
lf(theme('noun(161082-1-5)',obj('noun(161082-1-4)'))).
lf(source('noun(161082-1-5)',from('noun(161082-1-7)'))).
lf(location('noun(161082-1-5)',in('noun(161082-1-a24)'))).
lf(activity('noun(161082-1-5)',launch)).
lf(attribute('noun(161082-2-3)',isa('noun(161082-2-a23)'))).
lf(inst('noun(161082-2-3)',target)).
::::::::::::::
163030
```

```
::::::::::::::

lf(theme('noun(163030-2-1)',obj('noun(163030-1-2)'))).
lf(attribute('noun(163030-2-1)',front)).
lf(quantity('noun(163030-2-1)','3/4')).
lf(theme('noun(163030-2-1)',obj('noun(163030-2-2)'))).
lf(inst('noun(163030-2-1)','overall view')).
lf(attribute('noun(163030-1-a25)',part_of('noun(163030-1-a26)'))).
lf(inst('noun(163030-1-a25)','VMA-513')).
lf(quantity('noun(163030-1-2)',plural)).
lf(location('noun(163030-1-2)',on('noun(163030-1-8)'))).
lf(inst('noun(163030-1-2)','FAE')).
lf(attribute('noun(163030-1-8)',owned_by('noun(163030-1-a25)'))).
lf(attribute('noun(163030-1-8)',has_part('noun(163030-1-9)'))).
lf(attribute('noun(163030-1-8)',inst('noun(163030-1-10)'))).
lf(attribute('noun(163030-1-8)',inst('noun(163030-1-6)'))).
lf(inst('noun(163030-1-8)','AV-8A')).
lf(location('noun(163030-1-10)',on('noun(163030-1-11)'))).
lf(inst('noun(163030-1-10)','WF')).
lf(attribute('noun(163030-1-6)','158389')).
lf(inst('noun(163030-1-6)',bureau_no)).
lf(inst('noun(163030-2-3)',background)).
lf(attribute('noun(163030-2-2)','1')).
lf(location('noun(163030-2-2)',in('noun(163030-2-3)'))).
lf(inst('noun(163030-2-2)',hangar)).
lf(inst('noun(163030-1-a26)','USMC')).
lf(inst('noun(163030-1-11)',tail)).
lf(attribute('noun(163030-1-9)','6')).
lf(inst('noun(163030-1-9)',nose)).
::::::::::::::
164803
::::::::::::::

lf(attribute('noun(164803-1-5)',has_part('noun(164803-1-6)'))).
lf(attribute('noun(164803-1-5)',inst('noun(164803-1-7)'))).
lf(attribute('noun(164803-1-5)',inst('noun(164803-1-4)'))).
lf(inst('noun(164803-1-5)','A-7E')).
lf(location('noun(164803-1-7)',on('noun(164803-1-8)'))).
lf(inst('noun(164803-1-7)','AJ')).
lf(attribute('noun(164803-1-4)','157525')).
lf(inst('noun(164803-1-4)',bureau_no)).
lf(theme('noun(164803-2-1)',obj('noun(164803-1-2)'))).
lf(quantity('noun(164803-2-1)',closeup)).
lf(inst('noun(164803-2-1)',view)).
lf(location('noun(164803-1-2)',on('noun(164803-1-5)'))).
lf(inst('noun(164803-1-2)','Shrike')).
lf(inst('noun(164803-1-8)',tail)).
lf(attribute('noun(164803-1-6)','306')).
lf(inst('noun(164803-1-6)',nose)).
::::::::::::::
164804
::::::::::::::

lf(theme('noun(164804-2-1)',obj('noun(164804-1-2)'))).
lf(inst('noun(164804-2-1)','overall view')).
lf(attribute('noun(164804-1-5)',has_part('noun(164804-1-6)'))).
lf(attribute('noun(164804-1-5)',inst('noun(164804-1-7)'))).
lf(attribute('noun(164804-1-5)',inst('noun(164804-1-4)'))).
lf(inst('noun(164804-1-5)','A-7E')).
lf(location('noun(164804-1-7)',on('noun(164804-1-8)'))).
lf(inst('noun(164804-1-7)','AJ')).
```

```
lf(attribute('noun(164804-1-4)','157525')).
lf(inst('noun(164804-1-4)',bureau_no)).
lf(location('noun(164804-1-2)',on('noun(164804-1-5)'))).
lf(inst('noun(164804-1-2)','Shrike')).
lf(inst('noun(164804-1-8)',tail)).
lf(attribute('noun(164804-1-6)','306')).
lf(inst('noun(164804-1-6)',nose)).
::::::::::::::
166318
::::::::::::::

lf(inst('noun(166318-1-2)','FTV-2')).
lf(attribute('noun(166318-2-1)',excellent)).
lf(theme('noun(166318-2-1)',obj('noun(166318-1-2)'))).
lf(inst('noun(166318-2-1)',view)).
lf(agent('prespart(166318-1-1)',obj('noun(166318-1-2)'))).
lf(source('prespart(166318-1-1)',obj('noun(166318-1-4)'))).
lf(activity('prespart(166318-1-1)',depart)).
lf(theme('pverb(166318-3-a27)',obj('noun(166318-1-2)'))).
lf(theme('pverb(166318-3-a27)',obj('noun(166318-1-4)'))).
lf(activity('pverb(166318-3-a27)',free)).
lf(attribute('noun(166318-2-3)',part_of('noun(166318-1-2)'))).
lf(inst('noun(166318-2-3)',plume)).
lf(inst('noun(166318-1-4)',launcher)).
::::::::::::::
168579
::::::::::::::

lf(location('noun(168579-3-2)',on('noun(168579-3-a28)'))).
lf(inst('noun(168579-3-2)','human being')).
lf(inst('noun(168579-2-6)','A-7E')).
lf(location('noun(168579-2-4)',on('noun(168579-2-6)'))).
lf(inst('noun(168579-2-4)','data link pod')).
lf(location('noun(168579-1-2)',in('noun(168579-1-4)'))).
lf(location('noun(168579-1-2)',on('noun(168579-1-3)'))).
lf(inst('noun(168579-1-2)','Walleye II')).
lf(inst('noun(168579-1-3)','USS Kitty Hawk')).
lf(inst('noun(168579-3-a28)','flight deck')).
lf(quantity('noun(168579-2-1)',closeup)).
lf(theme('noun(168579-2-1)',obj('noun(168579-1-2)'))).
lf(theme('noun(168579-2-1)',obj('noun(168579-2-4)'))).
lf(theme('noun(168579-2-1)',obj('noun(168579-3-2)'))).
lf(inst('noun(168579-2-1)',view)).
lf(inst('noun(168579-1-4)','Gulf of Tonkin')).
::::::::::::::
174921
::::::::::::::

lf(theme('noun(174921-1-4)',obj('noun(174921-1-a29)'))).
lf(theme('noun(174921-1-4)',obj('noun(174921-1-3)'))).
lf(inst('noun(174921-1-4)','video view')).
lf(inst('noun(174921-2-a30)',sequence)).
lf(attribute('noun(174921-2-2)',inst('noun(174921-2-a30)'))).
lf(inst('noun(174921-2-2)',composite)).
lf(inst('noun(174921-1-a29)','Condor')).
lf(attribute('noun(174921-1-3)','1')).
lf(inst('noun(174921-1-3)','Dev-Assist')).
lf(theme('pres(174921-1-1)',obj('noun(174921-1-6)'))).
lf(activity('pres(174921-1-1)',explode)).
lf(attribute('noun(174921-1-6)',part_of('noun(174921-1-a29)'))).
lf(inst('noun(174921-1-6)',warhead)).
```

```
::::::::::::::
178012
::::::::::::::

lf(inst('noun(178012-2-a33)','MK-81')).
lf(attribute('noun(178012-2-5)','454692')).
lf(attribute('noun(178012-2-5)',inst('noun(178012-2-a32)'))).
lf(attribute('noun(178012-2-5)',electric)).
lf(theme('noun(178012-2-5)',obj('noun(178012-2-a33)'))).
lf(attribute('noun(178012-2-5)','SCO-96')).
lf(inst('noun(178012-2-5)','cookoff test')).
lf(theme('pastpart(178012-4-1)',obj('noun(178012-4-2)'))).
lf(destination('pastpart(178012-4-1)',to('noun(178012-2-a33)'))).
lf(activity('pastpart(178012-4-1)',assemble)).
lf(theme('noun(178012-3-1)',obj('noun(178012-2-a33)'))).
lf(theme('noun(178012-3-1)',obj('noun(178012-4-2)'))).
lf(time('noun(178012-3-1)',pretest)).
lf(inst('noun(178012-3-1)',view)).
lf(inst('noun(178012-1-a31)','Roseville')).
lf(quantity('noun(178012-4-2)',plural)).
lf(inst('noun(178012-4-2)',wire)).
lf(theme('pastpart(178012-3-1)',obj('noun(178012-2-a33)'))).
lf(activity('pastpart(178012-3-1)',conceal)).
lf(inst('noun(178012-2-a32)',heat)).
lf(theme('noun(178012-1-2)',obj('noun(178012-1-a31)'))).
lf(inst('noun(178012-1-2)',event)).
::::::::::::::
180657
::::::::::::::

lf(attribute('noun(180657-3-2)','44')).
lf(inst('noun(180657-3-2)',camera)).
lf(inst('noun(180657-2-a35)','AV-8A')).
lf(agent('prespart(180657-4-1)',obj('noun(180657-4-1)'))).
lf(source('prespart(180657-4-1)',obj('noun(180657-4-2)'))).
lf(activity('prespart(180657-4-1)',depart)).
lf(attribute('coordinate(180657-3-1)','4505 '' N x 34 '' E')).
lf(inst('coordinate(180657-3-1)',coordinate)).
lf(attribute('noun(180657-2-a34)',part_of('noun(180657-2-a35)'))).
lf(inst('noun(180657-2-a34)','escape system')).
lf(attribute('noun(180657-2-6)','4')).
lf(theme('noun(180657-2-6)',obj('noun(180657-2-5)'))).
lf(inst('noun(180657-2-6)',round)).
lf(theme('noun(180657-2-5)',obj('noun(180657-2-a34)'))).
lf(quantity('noun(180657-2-5)',keas('600'))).
lf(inst('noun(180657-2-5)',test)).
lf(attribute('noun(180657-3-1)',synchronous)).
lf(source('noun(180657-3-1)',from('noun(180657-3-2)'))).
lf(location('noun(180657-3-1)',at('coordinate(180657-3-1)'))).
lf(activity('noun(180657-3-1)',launch)).
lf(location('noun(180657-7-1)',in('noun(180657-6-2)'))).
lf(inst('noun(180657-7-1)',debris)).
lf(inst('noun(180657-6-2)',air)).
lf(state('noun(180657-6-1)',free)).
lf(location('noun(180657-6-1)',in('noun(180657-6-2)'))).
lf(inst('noun(180657-6-1)',parachute)).
lf(inst('noun(180657-5-a36)',seat)).
lf(agent('prespart(180657-5-1)',obj('noun(180657-5-2)'))).
lf(activity('prespart(180657-5-1)',heat)).
lf(quantity('noun(180657-5-2)',plural)).
lf(attribute('noun(180657-5-2)',part_of('noun(180657-5-a36)'))).
```

```
lf(inst('noun(180657-5-2)',rocket)).
lf(inst('noun(180657-4-2)',cockpit)).
lf(inst('noun(180657-4-1)',dummy)).
lf(attribute('noun(180657-1-1)','3923')).
lf(inst('noun(180657-1-1)','test plan')).
::::::::::::::
181709
::::::::::::::

lf(state('noun(181709-1-2)',modify)).
lf(location('noun(181709-1-2)',in('noun(181709-2-5)'))).
lf(attribute('noun(181709-1-2)',inst('noun(181709-2-3)'))).
lf(inst('noun(181709-1-2)','A-7E')).
lf(attribute('noun(181709-2-3)','156734')).
lf(inst('noun(181709-2-3)',bureau_no)).
lf(attribute('noun(181709-1-a37)',part_of('noun(181709-1-2)'))).
lf(inst('noun(181709-1-a37)','FLIR detector')).
lf(theme('pastpart(181709-2-1)',obj('noun(181709-2-1)'))).
lf(location('pastpart(181709-2-1)',on('noun(181709-1-2)'))).
lf(activity('pastpart(181709-2-1)',assemble)).
lf(attribute('noun(181709-2-5)','3')).
lf(inst('noun(181709-2-5)',hangar)).
lf(inst('noun(181709-2-1)',pod)).
::::::::::::::
181754
::::::::::::::

lf(inst('noun(181754-2-3)','human being')).
lf(quantity('noun(181754-2-1)',full)).
lf(theme('noun(181754-2-1)',obj('noun(181754-1-3)'))).
lf(inst('noun(181754-2-1)','side view')).
lf(inst('noun(181754-1-5)','A-6A')).
lf(theme('pastpart(181754-1-1)',obj('noun(181754-1-1)'))).
lf(location('pastpart(181754-1-1)',in('noun(181754-1-3)'))).
lf(activity('pastpart(181754-1-1)',assemble)).
lf(location('noun(181754-1-3)',on('noun(181754-1-5)'))).
lf(attribute('noun(181754-1-3)',next_to('noun(181754-2-3)'))).
lf(inst('noun(181754-1-3)',pod)).
lf(inst('noun(181754-1-1)','ESS')).
::::::::::::::
181761
::::::::::::::

lf(location('noun(181761-2-7)',at('noun(181761-4-3)'))).
lf(inst('noun(181761-2-7)','DODX')).
lf(attribute('noun(181761-2-8)',part_of('noun(181761-2-7)'))).
lf(inst('noun(181761-2-8)',clock)).
lf(theme('noun(181761-2-4)',obj('noun(181761-2-7)'))).
lf(attribute('noun(181761-2-4)','SCO-146')).
lf(theme('noun(181761-2-4)',obj('noun(181761-2-5)'))).
lf(inst('noun(181761-2-4)','cookoff test')).
lf(attribute('noun(181761-4-1)',outside)).
lf(theme('noun(181761-4-1)',obj('noun(181761-2-7)'))).
lf(inst('noun(181761-4-1)',view)).
lf(inst('noun(181761-2-6)',pallet)).
lf(attribute('noun(181761-2-5)',inert)).
lf(quantity('noun(181761-2-5)',plural('24'))).
lf(location('noun(181761-2-5)',in('noun(181761-2-7)'))).
lf(location('noun(181761-2-5)',on('noun(181761-2-6)'))).
lf(inst('noun(181761-2-5)',bomb)).
lf(inst('noun(181761-1-a38)','Roseville')).
```

```
lf(attribute('noun(181761-4-3)','63')).
lf(inst('noun(181761-4-3)',building)).
lf(attribute('noun(181761-3-1)',inside)).
lf(theme('noun(181761-3-1)',obj('noun(181761-2-7)'))).
lf(inst('noun(181761-3-1)',damage)).
lf(theme('noun(181761-1-2)',obj('noun(181761-1-a38)'))).
lf(inst('noun(181761-1-2)',event)).
::::::::::::::
182711
::::::::::::::

lf(inst('noun(182711-1-a39)','AIM-9L')).
lf(inst('noun(182711-1-8)','G Range')).
lf(theme('noun(182711-1-1)',obj('noun(182711-1-5)'))).
lf(inst('noun(182711-1-1)','air-to-air view')).
lf(quantity('noun(182711-2-2)',plural('100'))).
lf(inst('noun(182711-2-2)',feet)).
lf(inst('noun(182711-1-7)','QF-86')).
lf(inst('noun(182711-1-a40)',air)).
lf(attribute('noun(182711-1-5)','EXT-024')).
lf(theme('noun(182711-1-5)',obj('noun(182711-1-a39)'))).
lf(location('noun(182711-1-5)',over('noun(182711-1-8)'))).
lf(destination('noun(182711-1-5)',at('noun(182711-1-7)'))).
lf(location('noun(182711-1-5)',in('noun(182711-1-a40)'))).
lf(activity('noun(182711-1-5)',launch)).
lf(attribute('be(182711-2-1)',inst('noun(182711-2-2)'))).
lf(theme('be(182711-2-1)',obj('noun(182711-1-a39)'))).
lf(attribute('be(182711-2-1)',out)).
lf(activity('be(182711-2-1)',be)).
::::::::::::::
182712
::::::::::::::

lf(inst('noun(182712-1-a41)','AIM-9L')).
lf(inst('noun(182712-1-8)','G Range')).
lf(theme('noun(182712-1-1)',obj('noun(182712-1-5)'))).
lf(inst('noun(182712-1-1)','air-to-air view')).
lf(inst('noun(182712-1-7)','QF-86')).
lf(inst('noun(182712-1-a42)',air)).
lf(attribute('noun(182712-1-5)','EXT-024')).
lf(theme('noun(182712-1-5)',obj('noun(182712-1-a41)'))).
lf(location('noun(182712-1-5)',over('noun(182712-1-8)'))).
lf(destination('noun(182712-1-5)',at('noun(182712-1-7)'))).
lf(location('noun(182712-1-5)',in('noun(182712-1-a42)'))).
lf(activity('noun(182712-1-5)',launch)).
lf(agent('prespart(182712-2-1)',obj('noun(182712-1-a41)'))).
lf(theme('prespart(182712-2-1)',obj('noun(182712-2-2)'))).
lf(activity('prespart(182712-2-1)',hit)).
lf(inst('noun(182712-2-2)',nose)).
::::::::::::::
182713
::::::::::::::

lf(inst('noun(182713-1-a43)','AIM-9L')).
lf(inst('noun(182713-1-8)','G Range')).
lf(theme('noun(182713-1-1)',obj('noun(182713-1-5)'))).
lf(inst('noun(182713-1-1)','air-to-air view')).
lf(inst('noun(182713-1-7)','QF-86')).
lf(inst('noun(182713-1-a44)',air)).
lf(attribute('noun(182713-1-5)','EXT-024')).
lf(theme('noun(182713-1-5)',obj('noun(182713-1-a43)'))).
```

```
lf(location('noun(182713-1-5)',over('noun(182713-1-8)'))).
lf(destination('noun(182713-1-5)',at('noun(182713-1-7)'))).
lf(location('noun(182713-1-5)',in('noun(182713-1-a44)'))).
lf(activity('noun(182713-1-5)',launch)).
lf(theme('prespart(182713-2-1)',obj('noun(182713-1-7)'))).
lf(activity('prespart(182713-2-1)',explode)).
::::::::::::::
183531
::::::::::::::

lf(inst('noun(183531-1-6)','PCH-1')).
lf(attribute('noun(183531-3-2)',left)).
lf(theme('noun(183531-3-2)',obj('noun(183531-1-6)'))).
lf(attribute('noun(183531-3-2)',excellent)).
lf(time('noun(183531-3-2)',during('prespart(183531-2-1)'))).
lf(inst('noun(183531-3-2)','side view')).
lf(location('noun(183531-1-a45)',on('noun(183531-1-6)'))).
lf(quantity('noun(183531-1-a45)',1)).
lf(inst('noun(183531-1-a45)','Harpoon')).
lf(inst('noun(183531-1-7)','Puget Sound')).
lf(agent('prespart(183531-2-1)',obj('noun(183531-1-6)'))).
lf(theme('prespart(183531-2-1)',obj('noun(183531-1-a45)'))).
lf(location('prespart(183531-2-1)',in('noun(183531-1-7)'))).
lf(activity('prespart(183531-2-1)',launch)).
::::::::::::::
185854
::::::::::::::

lf(attribute('noun(185854-1-5)',inst('noun(185854-1-4)'))).
lf(inst('noun(185854-1-5)','UH-1N')).
lf(attribute('noun(185854-1-4)','158771')).
lf(inst('noun(185854-1-4)',bureau_no)).
lf(theme('noun(185854-2-1)',obj('noun(185854-1-2)'))).
lf(quantity('noun(185854-2-1)',closeup)).
lf(inst('noun(185854-2-1)',view)).
lf(inst('noun(185854-1-a46)','rotary wing dispenser')).
lf(attribute('noun(185854-1-2)',part_of('noun(185854-1-a46)'))).
lf(location('noun(185854-1-2)',on('noun(185854-1-5)'))).
lf(inst('noun(185854-1-2)','spin canister')).
::::::::::::::
185860
::::::::::::::

lf(attribute('noun(185860-1-5)',inst('noun(185860-1-4)'))).
lf(inst('noun(185860-1-5)','UH-1N')).
lf(theme('noun(185860-2-1)',obj('noun(185860-1-5)'))).
lf(inst('noun(185860-2-1)','overall view')).
lf(attribute('noun(185860-1-4)','158771')).
lf(inst('noun(185860-1-4)',bureau_no)).
lf(inst('noun(185860-1-a47)','rotary wing dispenser')).
lf(attribute('noun(185860-1-2)',part_of('noun(185860-1-a47)'))).
lf(location('noun(185860-1-2)',on('noun(185860-1-5)'))).
lf(quantity('noun(185860-1-2)',plural)).
lf(attribute('noun(185860-1-2)',part_of('noun(185860-1-5)'))).
lf(inst('noun(185860-1-2)','spin canister')).
::::::::::::::
185864
::::::::::::::

lf(attribute('noun(185864-1-5)',inst('noun(185864-1-4)'))).
lf(inst('noun(185864-1-5)','UH-1N')).
```

```
lf(location('noun(185864-2-2)',on('noun(185864-1-5)'))).
lf(inst('noun(185864-2-2)','R. W. Strongback')).
lf(attribute('noun(185864-1-4)','158771')).
lf(inst('noun(185864-1-4)',bureau_no)).
lf(theme('noun(185864-2-1)',obj('noun(185864-2-2)'))).
lf(inst('noun(185864-2-1)',view)).
lf(inst('noun(185864-1-a48)','rotary wing dispenser')).
lf(attribute('noun(185864-1-2)',part_of('noun(185864-1-a48)'))).
lf(location('noun(185864-1-2)',on('noun(185864-1-5)'))).
lf(inst('noun(185864-1-2)','spin canister')).
::::::::::::::
185866
::::::::::::::

lf(attribute('noun(185866-1-5)',inst('noun(185866-1-4)'))).
lf(inst('noun(185866-1-5)','UH-1N')).
lf(attribute('noun(185866-1-4)','158771')).
lf(inst('noun(185866-1-4)',bureau_no)).
lf(theme('noun(185866-2-1)',obj('noun(185866-1-2)'))).
lf(inst('noun(185866-2-1)',view)).
lf(inst('noun(185866-1-a49)','rotary wing dispenser')).
lf(attribute('noun(185866-1-2)',part_of('noun(185866-1-a49)'))).
lf(location('noun(185866-1-2)',on('noun(185866-1-5)'))).
lf(attribute('noun(185866-1-2)','FMC')).
lf(inst('noun(185866-1-2)','spin canister')).
::::::::::::::
188716
::::::::::::::

lf(inst('noun(188716-2-4)','XN-01')).
lf(theme('noun(188716-2-5)',obj('noun(188716-2-4)'))).
lf(inst('noun(188716-2-5)','front view')).
lf(location('noun(188716-2-1)',on('noun(188716-2-4)'))).
lf(inst('noun(188716-2-1)','load test')).
lf(quantity('noun(188716-2-7)',degree('45'))).
lf(theme('noun(188716-2-7)',obj('noun(188716-2-4)'))).
lf(inst('noun(188716-2-7)',elevation)).
lf(attribute('noun(188716-1-1)',echo)).
lf(inst('noun(188716-1-1)',program)).
::::::::::::::
209362
::::::::::::::

lf(attribute('noun(209362-2-a51)',owned_by('noun(209362-2-a52)'))).
lf(inst('noun(209362-2-a51)','Assault Breaker')).
lf(theme('noun(209362-2-4)',obj('noun(209362-2-a50)'))).
lf(theme('noun(209362-2-4)',obj('noun(209362-2-a51)'))).
lf(inst('noun(209362-2-4)','sled test')).
lf(quantity('noun(209362-4-1)',plural)).
lf(inst('noun(209362-4-1)',missile)).
lf(attribute('coordinate(209362-3-1)','166 '' N x 35 '' W')).
lf(inst('coordinate(209362-3-1)',coordinate)).
lf(inst('noun(209362-2-a52)','Martin Marietta')).
lf(attribute('noun(209362-3-1)',synchronous)).
lf(location('noun(209362-3-1)',at('coordinate(209362-3-1)'))).
lf(activity('noun(209362-3-1)',launch)).
lf(theme('pastpart(209362-4-1)',obj('noun(209362-4-1)'))).
lf(activity('pastpart(209362-4-1)',release)).
lf(inst('noun(209362-2-a50)',warhead)).
lf(attribute('noun(209362-1-1)','9937')).
lf(inst('noun(209362-1-1)','test plan')).
```

```
::::::::::::::
209862
::::::::::::::

lf(attribute('noun(209862-2-3)',owned_by('noun(209862-1-a53)'))).
lf(location('noun(209862-2-3)',on('noun(209862-1-5)'))).
lf(location('noun(209862-2-3)',in('noun(209862-1-a54)'))).
lf(inst('noun(209862-2-3)','A-4M')).
lf(theme('noun(209862-4-1)',obj('noun(209862-4-2)'))).
lf(inst('noun(209862-4-1)',bank)).
lf(quantity('noun(209862-2-4)',plural)).
lf(location('noun(209862-2-4)',in('noun(209862-1-a54)'))).
lf(inst('noun(209862-2-4)','LGB')).
lf(quantity('noun(209862-2-2)',plural)).
lf(attribute('noun(209862-2-2)',owned_by('noun(209862-1-a53)'))).
lf(location('noun(209862-2-2)',on('noun(209862-1-5)'))).
lf(location('noun(209862-2-2)',in('noun(209862-1-a54)'))).
lf(inst('noun(209862-2-2)','A-7E')).
lf(attribute('noun(209862-2-1)',owned_by('noun(209862-1-a53)'))).
lf(location('noun(209862-2-1)',on('noun(209862-1-5)'))).
lf(location('noun(209862-2-1)',in('noun(209862-1-a54)'))).
lf(inst('noun(209862-2-1)','A-6E')).
lf(inst('noun(209862-1-a53)','VX-5')).
lf(theme('noun(209862-1-5)',obj('noun(209862-2-3)'))).
lf(theme('noun(209862-1-5)',obj('noun(209862-2-2)'))).
lf(theme('noun(209862-1-5)',obj('noun(209862-2-1)'))).
lf(inst('noun(209862-1-5)','Sinkex')).
lf(theme('noun(209862-1-1)',obj('noun(209862-2-3)'))).
lf(theme('noun(209862-1-1)',obj('noun(209862-2-2)'))).
lf(theme('noun(209862-1-1)',obj('noun(209862-2-1)'))).
lf(theme('noun(209862-1-1)',obj('noun(209862-2-4)'))).
lf(attribute('noun(209862-1-1)',excellent)).
lf(location('noun(209862-1-1)',over('noun(209862-3-3)'))).
lf(inst('noun(209862-1-1)','air-to-air view')).
lf(inst('noun(209862-4-3)',background)).
lf(inst('noun(209862-3-3)',desert)).
lf(agent('prespart(209862-2-1)',obj('noun(209862-2-1)'))).
lf(agent('prespart(209862-2-1)',obj('noun(209862-2-2)'))).
lf(agent('prespart(209862-2-1)',obj('noun(209862-2-3)'))).
lf(theme('prespart(209862-2-1)',obj('noun(209862-2-4)'))).
lf(activity('prespart(209862-2-1)',carry)).
lf(inst('noun(209862-1-a54)',air)).
lf(quantity('noun(209862-4-2)',plural)).
lf(location('noun(209862-4-2)',in('noun(209862-4-3)'))).
lf(inst('noun(209862-4-2)',cloud)).
::::::::::::::
210192
::::::::::::::

lf(attribute('noun(210192-1-a8)',part_of('noun(210192-1-a7)'))).
lf(inst('noun(210192-1-a8)','DSU-28B')).
lf(attribute('noun(210192-3-4)',owned_by('noun(210192-3-a9)'))).
lf(attribute('noun(210192-3-4)',has_part('noun(210192-3-5)'))).
lf(attribute('noun(210192-3-4)',inst('noun(210192-3-3)'))).
lf(inst('noun(210192-3-4)','A-3B')).
lf(attribute('noun(210192-3-3)','142667')).
lf(inst('noun(210192-3-3)',bureau_no)).
lf(theme('noun(210192-2-2)',obj('noun(210192-1-5)'))).
lf(inst('noun(210192-2-2)','flyover test')).
lf(inst('noun(210192-1-a7)','Phoenix')).
lf(attribute('noun(210192-1-5)',part_of('noun(210192-1-a7)'))).
```

```
lf(inst('noun(210192-1-5)','EX-62')).
lf(agent('noun(210192-3-6)',obj('noun(210192-3-4)'))).
lf(activity('noun(210192-3-6)',fly)).
lf(theme('noun(210192-4-1)',obj('noun(210192-3-4)'))).
lf(source('noun(210192-4-1)',from('noun(210192-4-2)'))).
lf(inst('noun(210192-4-1)',view)).
lf(inst('noun(210192-4-2)',land)).
lf(inst('noun(210192-3-a9)','PMTC')).
lf(attribute('noun(210192-3-5)','71')).
lf(inst('noun(210192-3-5)',nose)).
::::::::::::::
210455
::::::::::::::

lf(attribute('noun(210455-1-5)',inst('noun(210455-1-6)'))).
lf(attribute('noun(210455-1-5)',inst('noun(210455-1-3)'))).
lf(location('noun(210455-1-5)',in('noun(210455-1-a61)'))).
lf(inst('noun(210455-1-5)','A-4M')).
lf(quantity('noun(210455-1-12)',plural(2))).
lf(attribute('noun(210455-1-12)',inst('noun(210455-1-a58)'))).
lf(attribute('noun(210455-1-12)',owned_by('noun(210455-1-a59)'))).
lf(location('noun(210455-1-12)',on('noun(210455-1-5)'))).
lf(inst('noun(210455-1-12)','AGM-65C')).
lf(theme('noun(210455-3-1)',obj('noun(210455-1-5)'))).
lf(inst('noun(210455-3-1)','side view')).
lf(attribute('noun(210455-1-6)',2)).
lf(inst('noun(210455-1-6)','Marines')).
lf(attribute('noun(210455-1-3)','160264')).
lf(inst('noun(210455-1-3)',bureau_no)).
lf(theme('noun(210455-1-1)',obj('noun(210455-1-5)'))).
lf(inst('noun(210455-1-1)','air-to-air view')).
lf(agent('prespart(210455-2-1)',obj('noun(210455-1-5)'))).
lf(location('prespart(210455-2-1)',over('noun(210455-2-2)'))).
lf(activity('prespart(210455-2-1)',fly)).
lf(quantity('noun(210455-2-2)',plural)).
lf(attribute('noun(210455-2-2)',high)).
lf(inst('noun(210455-2-2)','Sierra Nevada')).
lf(inst('noun(210455-1-a59)','USAF')).
lf(inst('noun(210455-1-a58)',education)).
lf(inst('noun(210455-1-a61)',air)).
lf(attribute('noun(210455-1-a60)',part_of('noun(210455-1-12)'))).
lf(inst('noun(210455-1-a60)',laser)).
::::::::::::::
210593
::::::::::::::

lf(location('noun(210593-1-3)',at('noun(210593-1-4)'))).
lf(inst('noun(210593-1-3)','Trident')).
lf(attribute('noun(210593-1-4)',eastern)).
lf(inst('noun(210593-1-4)','test range')).
lf(theme('noun(210593-1-1)',obj('noun(210593-1-3)'))).
lf(time('noun(210593-1-1)','1979')).
lf(activity('noun(210593-1-1)',launch)).
::::::::::::::
213528
::::::::::::::

lf(theme('noun(213528-2-1)',obj('noun(213528-1-a63)'))).
lf(inst('noun(213528-2-1)','front view')).
lf(attribute('noun(213528-1-a63)',owned_by('noun(213528-1-a64)'))).
lf(inst('noun(213528-1-a63)','RCC')).
```

```
lf(theme('noun(213528-1-a62)',obj('noun(213528-1-a63)'))).
lf(inst('noun(213528-1-a62)',construction)).
lf(attribute('noun(213528-2-a65)',part_of('noun(213528-2-4)'))).
lf(inst('noun(213528-2-a65)',antenna)).
lf(theme('pastpart(213528-2-1)',obj('noun(213528-2-4)'))).
lf(activity('pastpart(213528-2-1)',complete)).
lf(location('noun(213528-2-4)',at('noun(213528-1-a63)'))).
lf(inst('noun(213528-2-4)',tower)).
lf(inst('noun(213528-1-a64)','NWC')).
lf(attribute('noun(213528-1-4)',inst('noun(213528-1-a62)'))).
lf(inst('noun(213528-1-4)',progress)).
::::::::::::::
213529
::::::::::::::

lf(theme('noun(213529-2-1)',obj('noun(213529-1-a67)'))).
lf(inst('noun(213529-2-1)','side view')).
lf(attribute('noun(213529-1-a67)',owned_by('noun(213529-1-a68)'))).
lf(inst('noun(213529-1-a67)','RCC')).
lf(theme('noun(213529-1-a66)',obj('noun(213529-1-a67)'))).
lf(inst('noun(213529-1-a66)',construction)).
lf(attribute('noun(213529-2-a69)',part_of('noun(213529-2-4)'))).
lf(inst('noun(213529-2-a69)',antenna)).
lf(theme('pastpart(213529-2-1)',obj('noun(213529-2-4)'))).
lf(activity('pastpart(213529-2-1)',complete)).
lf(location('noun(213529-2-4)',at('noun(213529-1-a67)'))).
lf(inst('noun(213529-2-4)',tower)).
lf(inst('noun(213529-1-a68)','NWC')).
lf(attribute('noun(213529-1-4)',inst('noun(213529-1-a66)'))).
lf(inst('noun(213529-1-4)',progress)).
::::::::::::::
213795
::::::::::::::

lf(attribute('noun(213795-1-5)',inst('noun(213795-1-4)'))).
lf(inst('noun(213795-1-5)','UH-1N')).
lf(theme('noun(213795-2-1)',obj('noun(213795-2-3)'))).
lf(inst('noun(213795-2-1)','inside view')).
lf(inst('noun(213795-1-4)','Misty 13')).
lf(attribute('noun(213795-1-2)','Hytas')).
lf(location('noun(213795-1-2)',on('noun(213795-1-5)'))).
lf(inst('noun(213795-1-2)','flight test')).
lf(attribute('noun(213795-2-3)',part_of('noun(213795-1-5)'))).
lf(inst('noun(213795-2-3)',instrument)).
::::::::::::::
213798
::::::::::::::

lf(attribute('noun(213798-1-5)',inst('noun(213798-1-4)'))).
lf(inst('noun(213798-1-5)','UH-1N')).
lf(inst('noun(213798-1-4)','Misty 13')).
lf(attribute('noun(213798-1-2)','Hytas')).
lf(location('noun(213798-1-2)',on('noun(213798-1-5)'))).
lf(inst('noun(213798-1-2)','flight test')).
lf(inst('noun(213798-3-4)',background)).
lf(location('noun(213798-3-3)',in('noun(213798-3-4)'))).
lf(inst('noun(213798-3-3)','airport tower')).
lf(inst('noun(213798-3-2)',hangar)).
lf(attribute('noun(213798-3-1)',excellent)).
lf(theme('noun(213798-3-1)',obj('noun(213798-3-2)'))).
lf(theme('noun(213798-3-1)',obj('noun(213798-3-3)'))).
```

```
lf(inst('noun(213798-3-1)',view)).
lf(agent('prespart(213798-2-1)',obj('noun(213798-1-5)'))).
lf(activity('prespart(213798-2-1)',depart)).
::::::::::::::
213799
::::::::::::::

lf(attribute('noun(213799-1-5)',inst('noun(213799-1-4)'))).
lf(inst('noun(213799-1-5)','UH-1N')).
lf(inst('noun(213799-1-4)','Misty 13')).
lf(attribute('noun(213799-1-2)','Hytas')).
lf(location('noun(213799-1-2)',on('noun(213799-1-5)'))).
lf(inst('noun(213799-1-2)','flight test')).
lf(attribute('noun(213799-2-1)',front)).
lf(quantity('noun(213799-2-1)',closeup)).
lf(theme('noun(213799-2-1)',obj('noun(213799-1-5)'))).
lf(inst('noun(213799-2-1)',view)).
::::::::::::::
213853
::::::::::::::

lf(inst('noun(213853-1-6)','Mark-7')).
lf(location('noun(213853-1-3)',on('noun(213853-1-6)'))).
lf(inst('noun(213853-1-3)','Skipper')).
::::::::::::::
213855
::::::::::::::

lf(inst('noun(213855-2-1)','human being')).
lf(destination('noun(213855-1-5)',to('noun(213855-1-8)'))).
lf(inst('noun(213855-1-5)','Mark-7')).
lf(attribute('noun(213855-1-8)',inst('noun(213855-1-9)'))).
lf(attribute('noun(213855-1-8)',inst('noun(213855-1-7)'))).
lf(inst('noun(213855-1-8)','A-7C')).
lf(location('noun(213855-1-9)',on('noun(213855-1-10)'))).
lf(inst('noun(213855-1-9)','CL')).
lf(attribute('noun(213855-1-7)','156739')).
lf(inst('noun(213855-1-7)',bureau_no)).
lf(theme('pastpart(213855-1-1)',obj('noun(213855-1-3)'))).
lf(source('pastpart(213855-1-1)',from('noun(213855-1-5)'))).
lf(agent('pastpart(213855-1-1)',obj('noun(213855-2-1)'))).
lf(location('pastpart(213855-1-1)',on('noun(213855-1-8)'))).
lf(activity('pastpart(213855-1-1)',assemble)).
lf(inst('noun(213855-1-3)','Skipper')).
lf(inst('noun(213855-1-10)',tail)).
::::::::::::::
213856
::::::::::::::

lf(quantity('noun(213856-2-1)','1/4')).
lf(theme('noun(213856-2-1)',obj('noun(213856-1-6)'))).
lf(inst('noun(213856-2-1)','front view')).
lf(attribute('noun(213856-1-6)',inst('noun(213856-1-7)'))).
lf(attribute('noun(213856-1-6)',inst('noun(213856-1-5)'))).
lf(location('noun(213856-1-6)',on('noun(213856-2-3)'))).
lf(inst('noun(213856-1-6)','A-7C')).
lf(location('noun(213856-1-7)',on('noun(213856-1-8)'))).
lf(inst('noun(213856-1-7)','CL')).
lf(attribute('noun(213856-1-5)','156739')).
lf(inst('noun(213856-1-5)',bureau_no)).
lf(location('noun(213856-1-3)',on('noun(213856-1-6)'))).
```

```
lf(inst('noun(213856-1-3)','Skipper')).
lf(inst('noun(213856-2-3)',runway)).
lf(inst('noun(213856-1-8)',tail)).
::::::::::::::
213857
::::::::::::::

lf(theme('noun(213857-2-1)',obj('noun(213857-1-6)'))).
lf(inst('noun(213857-2-1)','side view')).
lf(attribute('noun(213857-1-6)',inst('noun(213857-1-7)'))).
lf(attribute('noun(213857-1-6)',inst('noun(213857-1-5)'))).
lf(location('noun(213857-1-6)',on('noun(213857-2-3)'))).
lf(inst('noun(213857-1-6)','A-7C')).
lf(location('noun(213857-1-7)',on('noun(213857-1-8)'))).
lf(inst('noun(213857-1-7)','CL')).
lf(attribute('noun(213857-1-5)','156739')).
lf(inst('noun(213857-1-5)',bureau_no)).
lf(location('noun(213857-1-3)',on('noun(213857-1-6)'))).
lf(inst('noun(213857-1-3)','Skipper')).
lf(inst('noun(213857-2-3)',runway)).
lf(inst('noun(213857-1-8)',tail)).
::::::::::::::
215669
::::::::::::::

lf(inst('noun(215669-4-2)',sled)).
lf(inst('noun(215669-2-a71)','A-7B/E')).
lf(attribute('noun(215669-2-a70)',part_of('noun(215669-2-a71)'))).
lf(inst('noun(215669-2-a70)','DVT-7')).
lf(agent('prespart(215669-4-1)',obj('noun(215669-4-1)'))).
lf(source('prespart(215669-4-1)',obj('noun(215669-4-2)'))).
lf(activity('prespart(215669-4-1)',depart)).
lf(attribute('coordinate(215669-3-1)','1090 '' N x 38 '' W')).
lf(inst('coordinate(215669-3-1)',coordinate)).
lf(attribute('noun(215669-2-6)','2')).
lf(theme('noun(215669-2-6)',obj('noun(215669-2-5)'))).
lf(inst('noun(215669-2-6)',run)).
lf(theme('noun(215669-2-5)',obj('noun(215669-2-a70)'))).
lf(quantity('noun(215669-2-5)',keas('250'))).
lf(inst('noun(215669-2-5)',test)).
lf(attribute('noun(215669-3-1)',synchronous)).
lf(location('noun(215669-3-1)',at('coordinate(215669-3-1)'))).
lf(activity('noun(215669-3-1)',launch)).
lf(inst('noun(215669-4-1)',dummy)).
lf(attribute('noun(215669-1-1)','1314')).
lf(inst('noun(215669-1-1)','test plan')).
::::::::::::::
216382
::::::::::::::

lf(inst('noun(216382-1-5)','AIM-9M')).
lf(theme('noun(216382-2-1)',obj('noun(216382-1-a72)'))).
lf(theme('noun(216382-2-1)',obj('noun(216382-2-3)'))).
lf(inst('noun(216382-2-1)',view)).
lf(inst('noun(216382-1-a72)','QF-86')).
lf(inst('noun(216382-1-2)','F-15')).
lf(theme('noun(216382-2-3)',obj('noun(216382-1-a72)'))).
lf(activity('noun(216382-2-3)',hit)).
lf(theme('prespart(216382-1-1)',obj('noun(216382-1-5)'))).
lf(destination('prespart(216382-1-1)',against('noun(216382-1-7)'))).
lf(source('prespart(216382-1-1)',from('noun(216382-1-2)'))).
```

```
lf(activity('prespart(216382-1-1)',launch)).
lf(attribute('noun(216382-1-7)',isa('noun(216382-1-a72)'))).
lf(inst('noun(216382-1-7)',target)).
::::::::::::::
216383
::::::::::::::

lf(attribute('noun(216383-1-10)',has_part('noun(216383-1-11)'))).
lf(location('noun(216383-1-10)',on('noun(216383-1-4)'))).
lf(inst('noun(216383-1-10)','XAIM-54C')).
lf(attribute('noun(216383-1-4)',has_part('noun(216383-1-5)'))).
lf(attribute('noun(216383-1-4)',inst('noun(216383-1-6)'))).
lf(attribute('noun(216383-1-4)',inst('noun(216383-1-3)'))).
lf(location('noun(216383-1-4)',in('noun(216383-1-a73)'))).
lf(inst('noun(216383-1-4)','F-14A')).
lf(attribute('noun(216383-1-3)','157990')).
lf(inst('noun(216383-1-3)',bureau_no)).
lf(theme('noun(216383-1-1)',obj('noun(216383-1-4)'))).
lf(inst('noun(216383-1-1)','air-to-air view')).
lf(inst('noun(216383-1-11)','EDM-4')).
lf(inst('noun(216383-1-a73)',air)).
lf(inst('noun(216383-1-7)',tail)).
lf(location('noun(216383-1-6)',on('noun(216383-1-7)'))).
lf(inst('noun(216383-1-6)','PMTC')).
lf(attribute('noun(216383-1-5)','211')).
lf(inst('noun(216383-1-5)',nose)).
::::::::::::::
217938
::::::::::::::

lf(location('noun(217938-1-4)',over('noun(217938-1-8)'))).
lf(attribute('noun(217938-1-4)',inst('noun(217938-1-5)'))).
lf(attribute('noun(217938-1-4)',inst('noun(217938-1-3)'))).
lf(location('noun(217938-1-4)',in('noun(217938-1-a75)'))).
lf(inst('noun(217938-1-4)','F-4B')).
lf(attribute('noun(217938-1-3)','149451')).
lf(inst('noun(217938-1-3)',bureau_no)).
lf(theme('noun(217938-1-1)',obj('noun(217938-1-4)'))).
lf(inst('noun(217938-1-1)','air-to-air view')).
lf(inst('noun(217938-1-a74)','Sierra Nevada')).
lf(location('noun(217938-1-5)',on('noun(217938-1-6)'))).
lf(inst('noun(217938-1-5)','China Lake')).
lf(inst('noun(217938-1-a75)',air)).
lf(quantity('noun(217938-1-8)',plural)).
lf(attribute('noun(217938-1-8)',inst('noun(217938-1-a74)'))).
lf(inst('noun(217938-1-8)',mountain)).
lf(inst('noun(217938-1-6)',tail)).
::::::::::::::
217947
::::::::::::::

lf(attribute('noun(217947-2-a77)',owned_by('noun(217947-2-a78)'))).
lf(location('noun(217947-2-a77)',on('noun(217947-4-3)'))).
lf(inst('noun(217947-2-a77)','Assault Breaker')).
lf(agent('prespart(217947-5-1)',obj('noun(217947-5-1)'))).
lf(activity('prespart(217947-5-1)',show)).
lf(theme('noun(217947-4-1)',obj('noun(217947-2-a77)'))).
lf(inst('noun(217947-4-1)',view)).
lf(attribute('coordinate(217947-3-1)','3377 '' N x 24 '' E')).
lf(inst('coordinate(217947-3-1)',coordinate)).
lf(inst('noun(217947-2-a78)','Martin Marietta')).
```

```
lf(inst('noun(217947-2-a76)','CSWS dispenser')).
lf(attribute('noun(217947-2-5)','1')).
lf(theme('noun(217947-2-5)',obj('noun(217947-2-4)'))).
lf(inst('noun(217947-2-5)',round)).
lf(theme('noun(217947-2-4)',obj('noun(217947-2-a76)'))).
lf(theme('noun(217947-2-4)',obj('noun(217947-2-a77)'))).
lf(inst('noun(217947-2-4)',test)).
lf(attribute('noun(217947-3-1)',synchronous)).
lf(location('noun(217947-3-1)',at('coordinate(217947-3-1)'))).
lf(activity('noun(217947-3-1)',launch)).
lf(inst('noun(217947-5-3)',air)).
lf(quantity('noun(217947-5-2)',plural)).
lf(location('noun(217947-5-2)',in('noun(217947-5-3)'))).
lf(inst('noun(217947-5-2)',projectile)).
lf(inst('noun(217947-5-1)',plume)).
lf(inst('noun(217947-4-3)',track)).
lf(attribute('noun(217947-1-1)','1346')).
lf(inst('noun(217947-1-1)','test plan')).
::::::::::::::
218178
::::::::::::::

lf(inst('noun(218178-2-1)',smoke)).
lf(inst('noun(218178-3-3)',helicopter)).
lf(inst('noun(218178-3-2)','DD-851')).
lf(location('noun(218178-1-2)',in('noun(218178-1-3)'))).
lf(inst('noun(218178-1-2)','CVA-59')).
lf(agent('prespart(218178-2-1)',obj('noun(218178-2-1)'))).
lf(source('prespart(218178-2-1)',from('noun(218178-1-2)'))).
lf(activity('prespart(218178-2-1)',depart)).
lf(inst('noun(218178-1-3)','Gulf of Tonkin')).
lf(activity('infin(218178-3-1)',aid)).
lf(goal('prespart(218178-3-1)',activity('infin(218178-3-1)'))).
lf(theme('prespart(218178-3-1)',obj('noun(218178-3-2)'))).
lf(theme('prespart(218178-3-1)',obj('noun(218178-3-3)'))).
lf(activity('prespart(218178-3-1)',wait)).
::::::::::::::
218179
::::::::::::::

lf(inst('noun(218179-2-1)','human being')).
lf(location('noun(218179-1-2)',in('noun(218179-1-3)'))).
lf(inst('noun(218179-1-2)','CVA-59')).
lf(inst('noun(218179-2-3)','flight deck')).
lf(inst('noun(218179-1-3)','Gulf of Tonkin')).
lf(agent('prespart(218179-2-1)',obj('noun(218179-2-1)'))).
lf(theme('prespart(218179-2-1)',obj('noun(218179-2-2)'))).
lf(location('prespart(218179-2-1)',on('noun(218179-2-3)'))).
lf(activity('prespart(218179-2-1)',fight)).
lf(state('noun(218179-2-2)',burn)).
lf(inst('noun(218179-2-2)',aircraft)).
::::::::::::::
218183
::::::::::::::

lf(inst('noun(218183-2-1)','human being')).
lf(state('noun(218183-2-3)',smoulder)).
lf(inst('noun(218183-2-3)','RA-5C')).
lf(location('noun(218183-1-2)',in('noun(218183-1-3)'))).
lf(inst('noun(218183-1-2)','CVA-59')).
lf(theme('noun(218183-2-7)',obj('noun(218183-2-6)'))).
```

```
lf(inst('noun(218183-2-7)',spreading)).
lf(inst('noun(218183-2-5)','flight deck')).
lf(location('noun(218183-2-4)',on('noun(218183-2-5)'))).
lf(inst('noun(218183-2-4)',edge)).
lf(inst('noun(218183-1-3)','Gulf of Tonkin')).
lf(theme('infin(218183-2-1)',obj('noun(218183-2-6)'))).
lf(destination('infin(218183-2-1)',from('noun(218183-2-7)'))).
lf(activity('infin(218183-2-1)',cease)).
lf(goal('prespart(218183-2-1)',activity('infin(218183-2-1)'))).
lf(agent('prespart(218183-2-1)',obj('noun(218183-2-1)'))).
lf(theme('prespart(218183-2-1)',obj('noun(218183-2-3)'))).
lf(destination('prespart(218183-2-1)',to('noun(218183-2-4)'))).
lf(activity('prespart(218183-2-1)',ptrans)).
lf(inst('noun(218183-2-6)',fire)).
::::::::::::::
218184
::::::::::::::

lf(inst('noun(218184-2-1)','human being')).
lf(location('noun(218184-1-2)',in('noun(218184-1-3)'))).
lf(inst('noun(218184-1-2)','CVA-59')).
lf(inst('noun(218184-2-3)','flight deck')).
lf(inst('noun(218184-1-3)','Gulf of Tonkin')).
lf(agent('prespart(218184-2-1)',obj('noun(218184-2-1)'))).
lf(theme('prespart(218184-2-1)',obj('noun(218184-2-2)'))).
lf(location('prespart(218184-2-1)',on('noun(218184-2-3)'))).
lf(activity('prespart(218184-2-1)',fight)).
lf(inst('noun(218184-2-2)',fire)).
::::::::::::::
218185
::::::::::::::

lf(inst('noun(218185-2-1)','human being')).
lf(location('noun(218185-1-2)',in('noun(218185-1-3)'))).
lf(inst('noun(218185-1-2)','CVA-59')).
lf(agent('prespart(218185-2-1)',obj('noun(218185-2-1)'))).
lf(theme('prespart(218185-2-1)',obj('noun(218185-2-2)'))).
lf(location('prespart(218185-2-1)',on('noun(218185-2-3)'))).
lf(activity('prespart(218185-2-1)',inspect)).
lf(inst('noun(218185-2-3)','flight deck')).
lf(inst('noun(218185-1-3)','Gulf of Tonkin')).
lf(state('noun(218185-2-2)',destroy)).
lf(inst('noun(218185-2-2)',aircraft)).
::::::::::::::
218186
::::::::::::::

lf(state('noun(218186-2-1)',shock)).
lf(inst('noun(218186-2-1)','human being')).
lf(location('noun(218186-1-2)',in('noun(218186-1-3)'))).
lf(inst('noun(218186-1-2)','CVA-59')).
lf(theme('pastpart(218186-2-1)',obj('noun(218186-2-2)'))).
lf(location('pastpart(218186-2-1)',in('noun(218186-2-3)'))).
lf(activity('pastpart(218186-2-1)',assemble)).
lf(attribute('noun(218186-2-3)','3')).
lf(inst('noun(218186-2-3)','hangar bay')).
lf(inst('noun(218186-1-3)','Gulf of Tonkin')).
lf(agent('prespart(218186-2-1)',obj('noun(218186-2-1)'))).
lf(theme('prespart(218186-2-1)',obj('noun(218186-2-2)'))).
lf(activity('prespart(218186-2-1)',observe)).
lf(quantity('noun(218186-2-2)',plural)).
```

```
lf(state('noun(218186-2-2)',dead)).
lf(inst('noun(218186-2-2)',body)).
::::::::::::::
218187
::::::::::::::

lf(inst('noun(218187-2-1)','human being')).
lf(location('noun(218187-1-2)',in('noun(218187-1-3)'))).
lf(inst('noun(218187-1-2)','CVA-59')).
lf(agent('prespart(218187-2-1)',obj('noun(218187-2-1)'))).
lf(theme('prespart(218187-2-1)',obj('noun(218187-2-2)'))).
lf(location('prespart(218187-2-1)',on('noun(218187-2-3)'))).
lf(activity('prespart(218187-2-1)',inspect)).
lf(inst('noun(218187-2-3)','flight deck')).
lf(inst('noun(218187-1-3)','Gulf of Tonkin')).
lf(state('noun(218187-2-2)',burn)).
lf(inst('noun(218187-2-2)',remains)).
::::::::::::::
218188
::::::::::::::

lf(inst('noun(218188-2-a79)','John')).
lf(attribute('noun(218188-2-3)',inst('noun(218188-2-a79)'))).
lf(attribute('noun(218188-2-3)',inst('noun(218188-2-a80)'))).
lf(attribute('noun(218188-2-3)',employee_of('noun(218188-2-4)'))).
lf(inst('noun(218188-2-3)','Beling')).
lf(location('noun(218188-1-2)',in('noun(218188-1-3)'))).
lf(inst('noun(218188-1-2)','CVA-59')).
lf(inst('noun(218188-2-a80)','Captain')).
lf(agent('prespart(218188-2-1)',obj('noun(218188-2-3)'))).
lf(theme('prespart(218188-2-1)',obj('noun(218188-2-5)'))).
lf(destination('prespart(218188-2-1)',to('noun(218188-2-6)'))).
lf(activity('prespart(218188-2-1)',inspect)).
lf(inst('noun(218188-2-6)','flight deck')).
lf(inst('noun(218188-2-4)','USN')).
lf(inst('noun(218188-1-3)','Gulf of Tonkin')).
lf(inst('noun(218188-2-5)',damage)).
::::::::::::::
218189
::::::::::::::

lf(inst('noun(218189-2-1)','human being')).
lf(location('noun(218189-1-2)',in('noun(218189-1-3)'))).
lf(inst('noun(218189-1-2)','CVA-59')).
lf(inst('noun(218189-1-3)','Gulf of Tonkin')).
lf(agent('prespart(218189-2-1)',obj('noun(218189-2-1)'))).
lf(theme('prespart(218189-2-1)',obj('noun(218189-2-2)'))).
lf(activity('prespart(218189-2-1)',ptrans)).
lf(state('noun(218189-2-2)',burn)).
lf(attribute('noun(218189-2-2)',bad)).
lf(inst('noun(218189-2-2)',aircraft)).
::::::::::::::
218915
::::::::::::::

lf(inst('noun(218915-2-a82)','A-7B/E')).
lf(attribute('noun(218915-2-a81)',part_of('noun(218915-2-a82)'))).
lf(inst('noun(218915-2-a81)','SRT-6')).
lf(attribute('coordinate(218915-3-1)','1075 '' N x 30 '' W')).
lf(inst('coordinate(218915-3-1)',coordinate)).
lf(theme('noun(218915-2-4)',obj('noun(218915-2-a81)'))).
```

```
lf(inst('noun(218915-2-4)',test)).
lf(attribute('noun(218915-3-1)',synchronous)).
lf(location('noun(218915-3-1)',at('coordinate(218915-3-1)'))).
lf(activity('noun(218915-3-1)',launch)).
lf(agent('prespart(218915-4-1)',obj('noun(218915-4-1)'))).
lf(theme('prespart(218915-4-1)',obj('noun(218915-4-2)'))).
lf(activity('prespart(218915-4-1)',injure)).
lf(inst('noun(218915-4-2)',canopy)).
lf(inst('noun(218915-4-1)',dummy)).
lf(attribute('noun(218915-1-1)','1356')).
lf(inst('noun(218915-1-1)','test plan')).
::::::::::::::
218997
::::::::::::::

lf(inst('noun(218997-2-a83)',strobe)).
lf(theme('noun(218997-1-3)',obj('noun(218997-1-2)'))).
lf(inst('noun(218997-1-3)','side view')).
lf(inst('noun(218997-1-2)','LWIR detector')).
lf(attribute('noun(218997-2-2)',s)).
lf(attribute('noun(218997-2-2)',inst('noun(218997-2-a83)'))).
lf(inst('noun(218997-2-2)',kind)).
lf(inst('noun(218997-1-4)',target)).
::::::::::::::
218999
::::::::::::::

lf(inst('noun(218999-1-a85)','MIG-21')).
lf(inst('noun(218999-2-a87)','mercury vapor')).
lf(attribute('noun(218999-2-a86)',isa('noun(218999-2-a87)'))).
lf(inst('noun(218999-2-a86)',luminaire)).
lf(theme('noun(218999-1-3)',obj('noun(218999-1-2)'))).
lf(theme('noun(218999-1-3)',obj('noun(218999-1-6)'))).
lf(inst('noun(218999-1-3)','side view')).
lf(inst('noun(218999-1-2)','LWIR detector')).
lf(attribute('noun(218999-2-3)',l)).
lf(attribute('noun(218999-2-3)',inst('noun(218999-2-a86)'))).
lf(inst('noun(218999-2-3)',kind)).
lf(inst('noun(218999-1-a84)',model)).
lf(attribute('noun(218999-1-6)',isa('noun(218999-1-a84)'))).
lf(attribute('noun(218999-1-6)',isa('noun(218999-1-a85)'))).
lf(inst('noun(218999-1-6)',target)).
::::::::::::::
219075
::::::::::::::

lf(attribute('noun(219075-2-1)',front)).
lf(quantity('noun(219075-2-1)','1/4')).
lf(theme('noun(219075-2-1)',obj('noun(219075-1-8)'))).
lf(inst('noun(219075-2-1)','overall view')).
lf(inst('noun(219075-1-a88)','VX-5')).
lf(attribute('noun(219075-1-8)',owned_by('noun(219075-1-a88)'))).
lf(attribute('noun(219075-1-8)',owned_by('noun(219075-1-a89)'))).
lf(attribute('noun(219075-1-8)',has_part('noun(219075-1-9)'))).
lf(attribute('noun(219075-1-8)',inst('noun(219075-1-10)'))).
lf(attribute('noun(219075-1-8)',inst('noun(219075-1-6)'))).
lf(inst('noun(219075-1-8)','AV-8A')).
lf(location('noun(219075-1-10)',on('noun(219075-1-11)'))).
lf(inst('noun(219075-1-10)','XE')).
lf(attribute('noun(219075-1-6)','158706')).
lf(inst('noun(219075-1-6)',bureau_no)).
```

```
lf(inst('noun(219075-1-a89)','USMC')).
lf(location('noun(219075-1-2)',on('noun(219075-1-8)'))).
lf(location('noun(219075-1-2)',on('noun(219075-2-5)'))).
lf(inst('noun(219075-1-2)','Sidearm')).
lf(attribute('noun(219075-2-a90)',right)).
lf(inst('noun(219075-2-a90)',wing)).
lf(attribute('noun(219075-2-5)',part_of('noun(219075-2-a90)'))).
lf(inst('noun(219075-2-5)',launcher)).
lf(inst('noun(219075-1-11)',tail)).
lf(attribute('noun(219075-1-9)','27')).
lf(inst('noun(219075-1-9)',nose)).
::::::::::::::
219079
::::::::::::::

lf(quantity('noun(219079-2-1)',overall)).
lf(theme('noun(219079-2-1)',obj('noun(219079-1-8)'))).
lf(inst('noun(219079-2-1)','rear view')).
lf(inst('noun(219079-1-a91)','VX-5')).
lf(attribute('noun(219079-1-8)',owned_by('noun(219079-1-a91)'))).
lf(attribute('noun(219079-1-8)',owned_by('noun(219079-1-a92)'))).
lf(attribute('noun(219079-1-8)',has_part('noun(219079-1-9)'))).
lf(attribute('noun(219079-1-8)',inst('noun(219079-1-10)'))).
lf(attribute('noun(219079-1-8)',inst('noun(219079-1-6)'))).
lf(inst('noun(219079-1-8)','AV-8A')).
lf(location('noun(219079-1-10)',on('noun(219079-1-11)'))).
lf(inst('noun(219079-1-10)','XE')).
lf(attribute('noun(219079-1-6)','158706')).
lf(inst('noun(219079-1-6)',bureau_no)).
lf(inst('noun(219079-1-a92)','USMC')).
lf(location('noun(219079-1-2)',on('noun(219079-1-8)'))).
lf(location('noun(219079-1-2)',on('noun(219079-2-5)'))).
lf(inst('noun(219079-1-2)','Sidearm')).
lf(attribute('noun(219079-2-a93)',right)).
lf(inst('noun(219079-2-a93)',wing)).
lf(attribute('noun(219079-2-5)',part_of('noun(219079-2-a93)'))).
lf(inst('noun(219079-2-5)',launcher)).
lf(inst('noun(219079-1-11)',tail)).
lf(attribute('noun(219079-1-9)','27')).
lf(inst('noun(219079-1-9)',nose)).
::::::::::::::
219539
::::::::::::::

lf(inst('noun(219539-2-a95)','J-52')).
lf(inst('noun(219539-4-a96)','dump test')).
lf(inst('noun(219539-1-1)','Aircraft Survivability Laboratory')).
lf(theme('noun(219539-2-3)',obj('noun(219539-2-a94)'))).
lf(inst('noun(219539-2-3)',ingestion)).
lf(attribute('noun(219539-4-2)',inst('noun(219539-4-a96)'))).
lf(inst('noun(219539-4-2)',setup)).
lf(attribute('noun(219539-3-1)','1380')).
lf(inst('noun(219539-3-1)','test plan')).
lf(attribute('noun(219539-2-a94)',consumed_by('noun(219539-2-a95)'))).
lf(inst('noun(219539-2-a94)',fuel)).
::::::::::::::
219551
::::::::::::::

lf(inst('noun(219551-2-1)','human being')).
lf(inst('noun(219551-1-4)','Michelson Laboratory')).
```

```
lf(activity('noun(219551-1-a98)',solder)).
lf(agent('prespart(219551-2-1)',obj('noun(219551-2-1)'))).
lf(theme('prespart(219551-2-1)',obj('noun(219551-2-2)'))).
lf(location('prespart(219551-2-1)',on('noun(219551-2-3)'))).
lf(activity('prespart(219551-2-1)',solder)).
lf(inst('noun(219551-2-3)',connector)).
lf(quantity('noun(219551-2-2)',plural)).
lf(inst('noun(219551-2-2)',wire)).
lf(inst('noun(219551-1-a97)',assembly)).
lf(attribute('noun(219551-1-3)',inst('noun(219551-1-a97)'))).
lf(attribute('noun(219551-1-3)',inst('noun(219551-1-a98)'))).
lf(location('noun(219551-1-3)',in('noun(219551-1-4)'))).
lf(inst('noun(219551-1-3)',location)).
::::::::::::::
219553
::::::::::::::

lf(inst('noun(219553-2-a102)','Linda')).
lf(attribute('noun(219553-2-2)',inst('noun(219553-2-a102)'))).
lf(inst('noun(219553-2-2)','Wincn')).
lf(inst('noun(219553-1-4)','Michelson Laboratory')).
lf(activity('noun(219553-1-a100)',solder)).
lf(inst('noun(219553-2-a101)','integrated circuit')).
lf(agent('prespart(219553-2-1)',obj('noun(219553-2-2)'))).
lf(theme('prespart(219553-2-1)',obj('noun(219553-2-4)'))).
lf(location('prespart(219553-2-1)',on('noun(219553-2-5)'))).
lf(activity('prespart(219553-2-1)',assemble)).
lf(inst('noun(219553-2-5)','solder board')).
lf(attribute('noun(219553-2-4)',isa('noun(219553-2-a101)'))).
lf(inst('noun(219553-2-4)',chip)).
lf(inst('noun(219553-1-a99)',assembly)).
lf(attribute('noun(219553-1-3)',inst('noun(219553-1-a99)'))).
lf(attribute('noun(219553-1-3)',inst('noun(219553-1-a100)'))).
lf(location('noun(219553-1-3)',in('noun(219553-1-4)'))).
lf(inst('noun(219553-1-3)',location)).
::::::::::::::
219554
::::::::::::::

lf(inst('noun(219554-2-1)','human being')).
lf(inst('noun(219554-1-4)','Michelson Laboratory')).
lf(activity('noun(219554-1-a104)',solder)).
lf(inst('noun(219554-2-a105)','G-R simulator')).
lf(agent('prespart(219554-2-1)',obj('noun(219554-2-1)'))).
lf(theme('prespart(219554-2-1)',obj('noun(219554-2-2)'))).
lf(location('prespart(219554-2-1)',on('noun(219554-2-4)'))).
lf(activity('prespart(219554-2-1)',solder)).
lf(attribute('noun(219554-2-4)',isa('noun(219554-2-a105)'))).
lf(inst('noun(219554-2-4)','circuit board')).
lf(inst('noun(219554-2-2)',resistor)).
lf(inst('noun(219554-1-a103)',assembly)).
lf(attribute('noun(219554-1-3)',inst('noun(219554-1-a103)'))).
lf(attribute('noun(219554-1-3)',inst('noun(219554-1-a104)'))).
lf(location('noun(219554-1-3)',in('noun(219554-1-4)'))).
lf(inst('noun(219554-1-3)',location)).
::::::::::::::
219555
::::::::::::::

lf(inst('noun(219555-2-1)','human being')).
lf(inst('noun(219555-1-4)','Michelson Laboratory')).
```

```
lf(activity('noun(219555-1-a107)',solder)).
lf(inst('noun(219555-2-a108)','G-R simulator')).
lf(theme('infin(219555-2-1)',obj('noun(219555-2-2)'))).
lf(location('infin(219555-2-1)',on('noun(219555-2-4)'))).
lf(activity('infin(219555-2-1)',solder)).
lf(attribute('noun(219555-2-4)',isa('noun(219555-2-a108)'))).
lf(inst('noun(219555-2-4)','circuit board')).
lf(inst('noun(219555-2-2)',resistor)).
lf(goal('prespart(219555-2-1)',activity('infin(219555-2-1)'))).
lf(agent('prespart(219555-2-1)',obj('noun(219555-2-1)'))).
lf(activity('prespart(219555-2-1)',prepare)).
lf(inst('noun(219555-1-a106)',assembly)).
lf(attribute('noun(219555-1-3)',inst('noun(219555-1-a106)'))).
lf(attribute('noun(219555-1-3)',inst('noun(219555-1-a107)'))).
lf(location('noun(219555-1-3)',in('noun(219555-1-4)'))).
lf(inst('noun(219555-1-3)',location)).
::::::::::::::
219557
::::::::::::::

lf(inst('noun(219557-2-a112)','Richard')).
lf(attribute('noun(219557-2-2)',inst('noun(219557-2-a112)'))).
lf(inst('noun(219557-2-2)','Maxwell')).
lf(inst('noun(219557-1-4)','Michelson Laboratory')).
lf(activity('noun(219557-1-a110)',solder)).
lf(inst('noun(219557-2-a111)','G-R simulator')).
lf(agent('prespart(219557-2-1)',obj('noun(219557-2-2)'))).
lf(theme('prespart(219557-2-1)',obj('noun(219557-2-3)'))).
lf(location('prespart(219557-2-1)',on('noun(219557-2-5)'))).
lf(activity('prespart(219557-2-1)',solder)).
lf(attribute('noun(219557-2-5)',isa('noun(219557-2-a111)'))).
lf(inst('noun(219557-2-5)','circuit board')).
lf(inst('noun(219557-2-3)',resistor)).
lf(inst('noun(219557-1-a109)',assembly)).
lf(attribute('noun(219557-1-3)',inst('noun(219557-1-a109)'))).
lf(attribute('noun(219557-1-3)',inst('noun(219557-1-a110)'))).
lf(location('noun(219557-1-3)',in('noun(219557-1-4)'))).
lf(inst('noun(219557-1-3)',location)).
::::::::::::::
219558
::::::::::::::

lf(inst('noun(219558-2-a115)','Linda')).
lf(attribute('noun(219558-2-7)',inst('noun(219558-2-a115)'))).
lf(inst('noun(219558-2-7)','Wincn')).
lf(inst('noun(219558-1-4)','Michelson Laboratory')).
lf(activity('noun(219558-1-a114)',solder)).
lf(theme('noun(219558-2-2)',obj('noun(219558-2-a116)'))).
lf(agent('noun(219558-2-2)',obj('noun(219558-2-7)'))).
lf(location('noun(219558-2-2)',in('noun(219558-1-3)'))).
lf(inst('noun(219558-2-2)',inspection)).
lf(inst('noun(219558-2-a116)',solder)).
lf(inst('noun(219558-1-a113)',assembly)).
lf(attribute('noun(219558-1-3)',inst('noun(219558-1-a113)'))).
lf(attribute('noun(219558-1-3)',inst('noun(219558-1-a114)'))).
lf(location('noun(219558-1-3)',in('noun(219558-1-4)'))).
lf(inst('noun(219558-1-3)',location)).
::::::::::::::
219907
::::::::::::::
```

```
lf(attribute('noun(219907-1-14)',next_to('noun(219907-1-15)'))).
lf(inst('noun(219907-1-14)','ALQ-119')).
lf(attribute('noun(219907-1-12)',owned_by('noun(219907-1-a118)'))).
lf(attribute('noun(219907-1-12)',next_to('noun(219907-1-14)'))).
lf(quantity('noun(219907-1-12)',plural)).
lf(location('noun(219907-1-12)',on('noun(219907-1-5)'))).
lf(inst('noun(219907-1-12)','AGM-65D')).
lf(attribute('noun(219907-1-5)',owned_by('noun(219907-1-a118)'))).
lf(attribute('noun(219907-1-5)',inst('noun(219907-1-3)'))).
lf(location('noun(219907-1-5)',in('noun(219907-1-a120)'))).
lf(inst('noun(219907-1-5)','F-4G')).
lf(quantity('noun(219907-1-17)',plural)).
lf(location('noun(219907-1-17)',on('noun(219907-1-5)'))).
lf(inst('noun(219907-1-17)','AGM-45')).
lf(attribute('noun(219907-1-8)',next_to('noun(219907-1-12)'))).
lf(quantity('noun(219907-1-8)',plural)).
lf(location('noun(219907-1-8)',on('noun(219907-1-5)'))).
lf(inst('noun(219907-1-8)','AGM-88')).
lf(attribute('noun(219907-1-3)','69263')).
lf(inst('noun(219907-1-3)','USAF_no')).
lf(theme('noun(219907-1-1)',obj('noun(219907-1-5)'))).
lf(inst('noun(219907-1-1)','air-to-air view')).
lf(inst('noun(219907-2-a119)','Sierra Nevada')).
lf(inst('noun(219907-2-3)',background)).
lf(inst('noun(219907-1-a118)','USAF')).
lf(attribute('noun(219907-1-a117)',part_of('noun(219907-1-12)'))).
lf(inst('noun(219907-1-a117)','infrared detector')).
lf(manner('pastpart(219907-1-1)',top_down('noun(219907-1-8)'))).
lf(theme('pastpart(219907-1-1)',obj('noun(219907-1-17)'))).
lf(theme('pastpart(219907-1-1)',obj('noun(219907-1-15)'))).
lf(theme('pastpart(219907-1-1)',obj('noun(219907-1-12)'))).
lf(theme('pastpart(219907-1-1)',obj('noun(219907-1-8)'))).
lf(location('pastpart(219907-1-1)',on('noun(219907-1-5)'))).
lf(activity('pastpart(219907-1-1)',assemble)).
lf(attribute('noun(219907-1-15)',next_to('noun(219907-1-17)'))).
lf(quantity('noun(219907-1-15)',plural)).
lf(location('noun(219907-1-15)',on('noun(219907-1-5)'))).
lf(inst('noun(219907-1-15)','Standard Arm')).
lf(inst('noun(219907-1-a120)',air)).
lf(quantity('noun(219907-2-2)',plural)).
lf(attribute('noun(219907-2-2)',inst('noun(219907-2-a119)'))).
lf(location('noun(219907-2-2)',in('noun(219907-2-3)'))).
lf(inst('noun(219907-2-2)',mountain)).
::::::::::::::
220152
::::::::::::::

lf(inst('noun(220152-1-a121)','Honeywell')).
lf(attribute('noun(220152-1-2)',owned_by('noun(220152-1-a121)'))).
lf(inst('noun(220152-1-2)',gun)).
lf(inst('noun(220152-2-4)',fireball)).
lf(quantity('noun(220152-2-2)',mm('120'))).
lf(location('noun(220152-2-2)',before('noun(220152-2-4)'))).
lf(quantity('noun(220152-2-2)',fps('3740'))).
lf(inst('noun(220152-2-2)',projectile)).
::::::::::::::
220748
::::::::::::::

lf(inst('noun(220748-2-a124)',pilot)).
lf(attribute('noun(220748-2-3)',inst('noun(220748-2-a123)'))).
```

```
lf(attribute('noun(220748-2-3)',inst('noun(220748-2-a124)'))).
lf(inst('noun(220748-2-3)','Kapernick')).
lf(attribute('noun(220748-1-10)',part_of('noun(220748-1-a122)'))).
lf(inst('noun(220748-1-10)','Coso Range')).
lf(inst('noun(220748-2-a123)','LCDR')).
lf(attribute('noun(220748-1-4)',inst('noun(220748-1-5)'))).
lf(attribute('noun(220748-1-4)',inst('noun(220748-1-3)'))).
lf(location('noun(220748-1-4)',in('noun(220748-1-a125)'))).
lf(inst('noun(220748-1-4)','A-7E')).
lf(quantity('noun(220748-1-8)',plural)).
lf(inst('noun(220748-1-8)','MK-82')).
lf(attribute('noun(220748-1-3)','160857')).
lf(inst('noun(220748-1-3)',bureau_no)).
lf(theme('noun(220748-1-1)',obj('noun(220748-1-4)'))).
lf(inst('noun(220748-1-1)','air-to-air view')).
lf(inst('noun(220748-1-11)',snow)).
lf(location('noun(220748-1-5)',on('noun(220748-1-6)'))).
lf(inst('noun(220748-1-5)','China Lake')).
lf(inst('noun(220748-1-a125)',air)).
lf(inst('noun(220748-1-a122)','NWC')).
lf(theme('pastpart(220748-1-1)',obj('noun(220748-1-10)'))).
lf(object('pastpart(220748-1-1)',with('noun(220748-1-11)'))).
lf(activity('pastpart(220748-1-1)',include)).
lf(theme('prespart(220748-1-1)',obj('noun(220748-1-8)'))).
lf(agent('prespart(220748-1-1)',obj('noun(220748-1-4)'))).
lf(location('prespart(220748-1-1)',over('noun(220748-1-10)'))).
lf(activity('prespart(220748-1-1)',launch)).
lf(inst('noun(220748-1-6)',tail)).
::::::::::::::
221353
::::::::::::::

lf(inst('noun(221353-2-a126)',tank)).
lf(attribute('noun(221353-2-7)',owned_by('noun(221353-2-a128)'))).
lf(inst('noun(221353-2-7)','F-4G')).
lf(inst('noun(221353-2-3)','modified AIM-9C')).
lf(agent('pres(221353-3-1)',obj('noun(221353-2-3)'))).
lf(destination('pres(221353-3-1)',obj('noun(221353-2-10)'))).
lf(activity('pres(221353-3-1)',approach)).
lf(theme('noun(221353-3-1)',obj('noun(221353-2-3)'))).
lf(inst('noun(221353-3-1)',view)).
lf(inst('noun(221353-2-a128)','USAF')).
lf(theme('noun(221353-2-4)',obj('noun(221353-2-3)'))).
lf(destination('noun(221353-2-4)',against('noun(221353-2-10)'))).
lf(source('noun(221353-2-4)',from('noun(221353-2-7)'))).
lf(activity('noun(221353-2-4)',launch)).
lf(inst('noun(221353-2-a127)',land)).
lf(attribute('noun(221353-2-11)',part_of('noun(221353-2-10)'))).
lf(inst('noun(221353-2-11)',radar)).
lf(attribute('noun(221353-2-10)',isa('noun(221353-2-a126)'))).
lf(location('noun(221353-2-10)',on('noun(221353-2-a127)'))).
lf(inst('noun(221353-2-10)',target)).
lf(attribute('noun(221353-1-1)','2162')).
lf(inst('noun(221353-1-1)','test plan')).
::::::::::::::
221354
::::::::::::::

lf(inst('noun(221354-2-a129)',tank)).
lf(attribute('noun(221354-2-7)',owned_by('noun(221354-2-a131)'))).
lf(inst('noun(221354-2-7)','F-4G')).
```

```
lf(inst('noun(221354-2-3)','modified AIM-9C')).
lf(agent('pres(221354-3-1)',obj('noun(221354-2-3)'))).
lf(destination('pres(221354-3-1)',obj('noun(221354-2-10)'))).
lf(activity('pres(221354-3-1)',arrive)).
lf(theme('noun(221354-3-1)',obj('noun(221354-2-3)'))).
lf(inst('noun(221354-3-1)',view)).
lf(inst('noun(221354-2-a131)','USAF')).
lf(theme('noun(221354-2-4)',obj('noun(221354-2-3)'))).
lf(destination('noun(221354-2-4)',against('noun(221354-2-10)'))).
lf(source('noun(221354-2-4)',from('noun(221354-2-7)'))).
lf(activity('noun(221354-2-4)',launch)).
lf(inst('noun(221354-2-a130)',land)).
lf(attribute('noun(221354-2-11)',part_of('noun(221354-2-10)'))).
lf(inst('noun(221354-2-11)',radar)).
lf(attribute('noun(221354-2-10)',isa('noun(221354-2-a129)'))).
lf(location('noun(221354-2-10)',on('noun(221354-2-a130)'))).
lf(inst('noun(221354-2-10)',target)).
lf(attribute('noun(221354-1-1)','2162')).
lf(inst('noun(221354-1-1)','test plan')).
::::::::::::::
223156
::::::::::::::

lf(inst('noun(223156-1-a132)',truck)).
lf(inst('noun(223156-1-6)','G-1 Range')).
lf(theme('noun(223156-2-1)',obj('noun(223156-1-2)'))).
lf(inst('noun(223156-2-1)',view)).
lf(agent('prespart(223156-1-1)',obj('noun(223156-1-2)'))).
lf(destination('prespart(223156-1-1)',into('noun(223156-1-5)'))).
lf(activity('prespart(223156-1-1)',approach)).
lf(time('noun(223156-1-2)',before('noun(223156-2-3)'))).
lf(inst('noun(223156-1-2)','Harm')).
lf(theme('noun(223156-2-3)',obj('noun(223156-1-2)'))).
lf(activity('noun(223156-2-3)',hit)).
lf(attribute('noun(223156-1-a133)',part_of('noun(223156-1-a132)'))).
lf(inst('noun(223156-1-a133)',radar)).
lf(attribute('noun(223156-1-5)',isa('noun(223156-1-a132)'))).
lf(location('noun(223156-1-5)',at('noun(223156-1-6)'))).
lf(inst('noun(223156-1-5)',target)).
::::::::::::::
224159
::::::::::::::

lf(inst('noun(224159-1-a134)',tank)).
lf(quantity('noun(224159-1-4)',plural)).
lf(attribute('noun(224159-1-4)',owned_by('noun(224159-1-a136)'))).
lf(inst('noun(224159-1-4)','AGM-65C')).
lf(inst('noun(224159-1-a136)','USAF')).
lf(attribute('noun(224159-1-a135)',part_of('noun(224159-1-4)'))).
lf(inst('noun(224159-1-a135)',laser)).
lf(agent('prespart(224159-1-1)',obj('noun(224159-1-4)'))).
lf(theme('prespart(224159-1-1)',obj('noun(224159-1-6)'))).
lf(location('prespart(224159-1-1)',at('noun(224159-1-7)'))).
lf(activity('prespart(224159-1-1)',hit)).
lf(inst('noun(224159-1-7)','Eglin AFB')).
lf(quantity('noun(224159-1-6)',plural)).
lf(attribute('noun(224159-1-6)',isa('noun(224159-1-a134)'))).
lf(inst('noun(224159-1-6)',target)).
::::::::::::::
224161
::::::::::::::
```

```
lf(inst('noun(224161-1-a137)',tank)).
lf(quantity('noun(224161-1-4)',plural)).
lf(attribute('noun(224161-1-4)',owned_by('noun(224161-1-a139)'))).
lf(inst('noun(224161-1-4)','AGM-65C')).
lf(theme('noun(224161-2-1)',obj('noun(224161-1-6)'))).
lf(inst('noun(224161-2-1)',view)).
lf(inst('noun(224161-1-a139)','USAF')).
lf(theme('prespart(224161-2-1)',obj('noun(224161-1-6)'))).
lf(activity('prespart(224161-2-1)',explode)).
lf(attribute('noun(224161-1-a138)',part_of('noun(224161-1-4)'))).
lf(inst('noun(224161-1-a138)',laser)).
lf(agent('prespart(224161-1-1)',obj('noun(224161-1-4)'))).
lf(theme('prespart(224161-1-1)',obj('noun(224161-1-6)'))).
lf(location('prespart(224161-1-1)',at('noun(224161-1-7)'))).
lf(activity('prespart(224161-1-1)',hit)).
lf(inst('noun(224161-1-7)','Eglin AFB')).
lf(quantity('noun(224161-1-6)',plural)).
lf(attribute('noun(224161-1-6)',isa('noun(224161-1-a137)'))).
lf(inst('noun(224161-1-6)',target)).
::::::::::::::
224163
::::::::::::::

lf(inst('noun(224163-1-a140)',tank)).
lf(quantity('noun(224163-1-4)',plural)).
lf(attribute('noun(224163-1-4)',owned_by('noun(224163-1-a142)'))).
lf(inst('noun(224163-1-4)','AGM-65C')).
lf(theme('noun(224163-2-3)',obj('noun(224163-1-6)'))).
lf(inst('noun(224163-2-3)',explosion)).
lf(theme('noun(224163-2-1)',obj('noun(224163-1-6)'))).
lf(inst('noun(224163-2-1)',view)).
lf(inst('noun(224163-1-a142)','USAF')).
lf(attribute('noun(224163-1-a141)',part_of('noun(224163-1-4)'))).
lf(inst('noun(224163-1-a141)',laser)).
lf(agent('prespart(224163-1-1)',obj('noun(224163-1-4)'))).
lf(theme('prespart(224163-1-1)',obj('noun(224163-1-6)'))).
lf(location('prespart(224163-1-1)',at('noun(224163-1-7)'))).
lf(time('prespart(224163-1-1)',after('noun(224163-2-3)'))).
lf(activity('prespart(224163-1-1)',hit)).
lf(inst('noun(224163-1-7)','Eglin AFB')).
lf(quantity('noun(224163-1-6)',plural)).
lf(attribute('noun(224163-1-6)',isa('noun(224163-1-a140)'))).
lf(inst('noun(224163-1-6)',target)).
::::::::::::::
224164
::::::::::::::

lf(inst('noun(224164-2-3)',truck)).
lf(inst('noun(224164-1-a143)',tank)).
lf(quantity('noun(224164-1-4)',plural)).
lf(attribute('noun(224164-1-4)',owned_by('noun(224164-1-a145)'))).
lf(inst('noun(224164-1-4)','AGM-65C')).
lf(theme('noun(224164-2-1)',obj('noun(224164-1-6)'))).
lf(theme('noun(224164-2-1)',obj('noun(224164-2-3)'))).
lf(inst('noun(224164-2-1)',view)).
lf(inst('noun(224164-1-a145)','USAF')).
lf(attribute('noun(224164-1-a144)',part_of('noun(224164-1-4)'))).
lf(inst('noun(224164-1-a144)',laser)).
lf(agent('prespart(224164-1-1)',obj('noun(224164-1-4)'))).
lf(theme('prespart(224164-1-1)',obj('noun(224164-1-6)'))).
```

```
lf(location('prespart(224164-1-1)',at('noun(224164-1-7)'))).
lf(activity('prespart(224164-1-1)',hit)).
lf(inst('noun(224164-1-7)','Eglin AFB')).
lf(quantity('noun(224164-1-6)',plural)).
lf(attribute('noun(224164-1-6)',isa('noun(224164-1-a143)'))).
lf(inst('noun(224164-1-6)',target)).
::::::::::::::
224547
::::::::::::::

lf(state('noun(224547-1-2)',section)).
lf(inst('noun(224547-1-2)','BLU-80/B')).
lf(inst('noun(224547-2-1)','tail fin')).
lf(theme('pastpart(224547-2-1)',obj('noun(224547-2-1)'))).
lf(activity('pastpart(224547-2-1)',close)).
lf(attribute('noun(224547-1-a146)',part_of('noun(224547-1-2)'))).
lf(attribute('noun(224547-1-a146)',binary)).
lf(inst('noun(224547-1-a146)','chemical element')).
::::::::::::::
224548
::::::::::::::

lf(state('noun(224548-1-2)',section)).
lf(inst('noun(224548-1-2)','BLU-80/B')).
lf(attribute('noun(224548-2-1)',rear)).
lf(inst('noun(224548-2-1)',opening)).
lf(attribute('noun(224548-1-a147)',part_of('noun(224548-1-2)'))).
lf(attribute('noun(224548-1-a147)',binary)).
lf(inst('noun(224548-1-a147)','chemical element')).
::::::::::::::
224549
::::::::::::::

lf(state('noun(224549-1-2)',section)).
lf(inst('noun(224549-1-2)','BLU-80/B')).
lf(attribute('noun(224549-2-2)',inst('noun(224549-2-a149)'))).
lf(inst('noun(224549-2-2)','latch assembly')).
lf(inst('noun(224549-2-a149)',tail)).
lf(theme('pastpart(224549-2-1)',obj('noun(224549-2-2)'))).
lf(activity('pastpart(224549-2-1)',close)).
lf(attribute('noun(224549-1-a148)',part_of('noun(224549-1-2)'))).
lf(attribute('noun(224549-1-a148)',binary)).
lf(inst('noun(224549-1-a148)','chemical element')).
::::::::::::::
224550
::::::::::::::

lf(state('noun(224550-1-2)',section)).
lf(inst('noun(224550-1-2)','BLU-80/B')).
lf(attribute('noun(224550-2-1)',front)).
lf(state('noun(224550-2-1)',pot)).
lf(inst('noun(224550-2-1)',opening)).
lf(attribute('noun(224550-1-a150)',part_of('noun(224550-1-2)'))).
lf(attribute('noun(224550-1-a150)',binary)).
lf(inst('noun(224550-1-a150)','chemical element')).
::::::::::::::
224551
::::::::::::::

lf(state('noun(224551-1-2)',section)).
lf(inst('noun(224551-1-2)','BLU-80/B')).
```

```
lf(theme('noun(224551-2-1)',obj('noun(224551-2-3)'))).
lf(inst('noun(224551-2-1)','total view')).
lf(attribute('noun(224551-2-3)',rear)).
lf(attribute('noun(224551-2-3)',part_of('noun(224551-1-2)'))).
lf(inst('noun(224551-2-3)',end)).
lf(theme('pastpart(224551-3-1)',obj('noun(224551-3-1)'))).
lf(activity('pastpart(224551-3-1)',open_act)).
lf(inst('noun(224551-3-1)',tail)).
lf(attribute('noun(224551-1-a151)',part_of('noun(224551-1-2)'))).
lf(attribute('noun(224551-1-a151)',binary)).
lf(inst('noun(224551-1-a151)','chemical element')).
::::::::::::::
224552
::::::::::::::

lf(state('noun(224552-1-2)',section)).
lf(inst('noun(224552-1-2)','BLU-80/B')).
lf(theme('noun(224552-2-1)',obj('noun(224552-2-3)'))).
lf(inst('noun(224552-2-1)','front view')).
lf(theme('pastpart(224552-3-1)',obj('noun(224552-3-1)'))).
lf(activity('pastpart(224552-3-1)',open_act)).
lf(inst('noun(224552-3-1)',tail)).
lf(attribute('noun(224552-2-3)',rear)).
lf(attribute('noun(224552-2-3)',part_of('noun(224552-1-2)'))).
lf(inst('noun(224552-2-3)',bulkhead)).
lf(attribute('noun(224552-1-a152)',part_of('noun(224552-1-2)'))).
lf(attribute('noun(224552-1-a152)',binary)).
lf(inst('noun(224552-1-a152)','chemical element')).
::::::::::::::
225194
::::::::::::::

lf(theme('noun(225194-3-1)',obj('noun(225194-2-2)'))).
lf(quantity('noun(225194-3-1)',full)).
lf(location('noun(225194-3-1)',over('noun(225194-3-2)'))).
lf(inst('noun(225194-3-1)','side view')).
lf(quantity('noun(225194-2-6)',plural)).
lf(attribute('noun(225194-2-6)',inst('noun(225194-2-a153)'))).
lf(attribute('noun(225194-2-6)',inert)).
lf(attribute('noun(225194-2-6)',part_of('noun(225194-2-2)'))).
lf(inst('noun(225194-2-6)','MK-82')).
lf(attribute('noun(225194-1-6)',owned_by('noun(225194-1-7)'))).
lf(attribute('noun(225194-1-6)',inst('noun(225194-1-7)'))).
lf(attribute('noun(225194-1-6)',inst('noun(225194-1-4)'))).
lf(location('noun(225194-1-6)',in('noun(225194-1-a155)'))).
lf(inst('noun(225194-1-6)','AV-8B')).
lf(attribute('noun(225194-1-7)','161398')).
lf(location('noun(225194-1-7)',on('noun(225194-1-8)'))).
lf(inst('noun(225194-1-7)','Marines')).
lf(attribute('noun(225194-1-4)','161398')).
lf(inst('noun(225194-1-4)',bureau_no)).
lf(theme('noun(225194-1-1)',obj('noun(225194-1-6)'))).
lf(inst('noun(225194-1-1)','air-to-air view')).
lf(inst('noun(225194-3-2)','Pax River')).
lf(inst('noun(225194-2-a154)',production)).
lf(inst('noun(225194-1-a155)',air)).
lf(inst('noun(225194-2-a153)',practice)).
lf(state('noun(225194-2-3)',camouflage)).
lf(attribute('noun(225194-2-3)',part_of('noun(225194-2-2)'))).
lf(inst('noun(225194-2-3)',paint)).
lf(attribute('noun(225194-2-2)',inst('noun(225194-2-a154)'))).
```

```
lf(inst('noun(225194-2-2)',model)).
lf(inst('noun(225194-1-8)',tail)).
::::::::::::::
225196
::::::::::::::

lf(theme('noun(225196-3-1)',obj('noun(225196-2-2)'))).
lf(quantity('noun(225196-3-1)','1/4')).
lf(location('noun(225196-3-1)',over('noun(225196-3-2)'))).
lf(location('noun(225196-3-1)',over('noun(225196-3-4)'))).
lf(inst('noun(225196-3-1)','overhead view')).
lf(quantity('noun(225196-2-6)',plural)).
lf(attribute('noun(225196-2-6)',inst('noun(225196-2-a156)'))).
lf(attribute('noun(225196-2-6)',inert)).
lf(attribute('noun(225196-2-6)',part_of('noun(225196-2-2)'))).
lf(inst('noun(225196-2-6)','MK-82')).
lf(attribute('noun(225196-1-6)',owned_by('noun(225196-1-7)'))).
lf(attribute('noun(225196-1-6)',inst('noun(225196-1-7)'))).
lf(attribute('noun(225196-1-6)',inst('noun(225196-1-4)'))).
lf(location('noun(225196-1-6)',in('noun(225196-1-a159)'))).
lf(inst('noun(225196-1-6)','AV-8B')).
lf(attribute('noun(225196-1-7)','161398')).
lf(location('noun(225196-1-7)',on('noun(225196-1-8)'))).
lf(inst('noun(225196-1-7)','Marines')).
lf(attribute('noun(225196-1-4)','161398')).
lf(inst('noun(225196-1-4)',bureau_no)).
lf(theme('noun(225196-1-1)',obj('noun(225196-1-6)'))).
lf(inst('noun(225196-1-1)','air-to-air view')).
lf(inst('noun(225196-3-2)','Missouri River')).
lf(inst('noun(225196-2-a157)',production)).
lf(inst('noun(225196-1-a159)',air)).
lf(inst('noun(225196-3-a158)',farm)).
lf(attribute('noun(225196-3-4)',inst('noun(225196-3-a158)'))).
lf(inst('noun(225196-3-4)',land)).
lf(inst('noun(225196-2-a156)',practice)).
lf(state('noun(225196-2-3)',camouflage)).
lf(attribute('noun(225196-2-3)',part_of('noun(225196-2-2)'))).
lf(inst('noun(225196-2-3)',paint)).
lf(attribute('noun(225196-2-2)',inst('noun(225196-2-a157)'))).
lf(inst('noun(225196-2-2)',model)).
lf(inst('noun(225196-1-8)',tail)).
::::::::::::::
226385
::::::::::::::

lf(inst('noun(226385-2-a160)','Jon')).
lf(location('noun(226385-2-5)',at('noun(226385-2-6)'))).
lf(inst('noun(226385-2-5)','F/A-18A simulator')).
lf(attribute('noun(226385-2-3)',inst('noun(226385-2-a160)'))).
lf(attribute('noun(226385-2-3)',inst('noun(226385-2-a161)'))).
lf(location('noun(226385-2-3)',at('noun(226385-2-4)'))).
lf(inst('noun(226385-2-3)','Gallinetti')).
lf(inst('noun(226385-2-a161)','Major')).
lf(inst('noun(226385-1-2)','F/A-18A')).
lf(attribute('noun(226385-2-7)','3')).
lf(inst('noun(226385-2-7)',hangar)).
lf(attribute('noun(226385-2-6)',isa('noun(226385-2-7)'))).
lf(inst('noun(226385-2-6)','WSSA')).
lf(quantity('noun(226385-2-4)',plural)).
lf(attribute('noun(226385-2-4)',part_of('noun(226385-2-5)'))).
lf(inst('noun(226385-2-4)','control equipment')).
```

```
::::::::::::::
226386
::::::::::::::

lf(inst('noun(226386-2-a162)','Jon')).
lf(location('noun(226386-2-5)',at('noun(226386-2-6)'))).
lf(inst('noun(226386-2-5)','F/A-18A simulator')).
lf(attribute('noun(226386-2-3)',inst('noun(226386-2-a162)'))).
lf(attribute('noun(226386-2-3)',inst('noun(226386-2-a163)'))).
lf(location('noun(226386-2-3)',at('noun(226386-2-4)'))).
lf(inst('noun(226386-2-3)','Gallinetti')).
lf(inst('noun(226386-2-a163)','Major')).
lf(inst('noun(226386-1-2)','F/A-18A')).
lf(theme('noun(226386-3-1)',obj('noun(226386-2-3)'))).
lf(inst('noun(226386-3-1)',view)).
lf(attribute('noun(226386-2-7)','3')).
lf(inst('noun(226386-2-7)',hangar)).
lf(attribute('noun(226386-2-6)',isa('noun(226386-2-7)'))).
lf(inst('noun(226386-2-6)','WSSA')).
lf(quantity('noun(226386-2-4)',plural)).
lf(attribute('noun(226386-2-4)',part_of('noun(226386-2-5)'))).
lf(inst('noun(226386-2-4)','control equipment')).
lf(theme('prespart(226386-3-1)',obj('noun(226386-2-3)'))).
lf(location('prespart(226386-3-1)',behind('noun(226386-2-3)'))).
lf(activity('prespart(226386-3-1)',inhabit)).
::::::::::::::
227282
::::::::::::::

lf(inst('noun(227282-2-5)','Aero 51A')).
lf(inst('noun(227282-2-2)','human being')).
lf(location('noun(227282-3-3)',in('noun(227282-3-6)'))).
lf(attribute('noun(227282-3-3)',inst('noun(227282-3-4)'))).
lf(attribute('noun(227282-3-3)',inst('noun(227282-3-2)'))).
lf(inst('noun(227282-3-3)','F-4J')).
lf(location('noun(227282-3-4)',on('noun(227282-3-5)'))).
lf(inst('noun(227282-3-4)','SH')).
lf(attribute('noun(227282-3-2)','158378')).
lf(inst('noun(227282-3-2)',bureau_no)).
lf(time('noun(227282-1-5)',preflight)).
lf(inst('noun(227282-1-5)','F/A-18A')).
lf(inst('noun(227282-3-6)',background)).
lf(theme('prespart(227282-2-1)',obj('noun(227282-1-3)'))).
lf(agent('prespart(227282-2-1)',obj('noun(227282-2-2)'))).
lf(source('prespart(227282-2-1)',from('noun(227282-2-5)'))).
lf(activity('prespart(227282-2-1)',disassemble)).
lf(theme('noun(227282-2-1)',obj('noun(227282-1-3)'))).
lf(theme('noun(227282-2-1)',obj('noun(227282-2-2)'))).
lf(inst('noun(227282-2-1)',view)).
lf(theme('pastpart(227282-1-1)',obj('noun(227282-1-3)'))).
lf(location('pastpart(227282-1-1)',on('noun(227282-1-5)'))).
lf(activity('pastpart(227282-1-1)',assemble)).
lf(quantity('noun(227282-1-3)',plural)).
lf(attribute('noun(227282-1-3)',isa('noun(227282-1-a164)'))).
lf(inst('noun(227282-1-3)','Samson')).
lf(inst('noun(227282-3-5)',tail)).
lf(inst('noun(227282-1-a164)',decoy)).
::::::::::::::
227283
::::::::::::::
```

```
lf(attribute('noun(227283-2-2)',inst('noun(227283-2-a166)'))).
lf(inst('noun(227283-2-2)','human being')).
lf(time('noun(227283-1-5)',preflight)).
lf(inst('noun(227283-1-5)','F/A-18A')).
lf(inst('noun(227283-2-a166)',ordnance)).
lf(theme('pastpart(227283-1-1)',obj('noun(227283-1-3)'))).
lf(location('pastpart(227283-1-1)',on('noun(227283-1-5)'))).
lf(agent('pastpart(227283-1-1)',obj('noun(227283-2-2)'))).
lf(activity('pastpart(227283-1-1)',assemble)).
lf(quantity('noun(227283-1-3)',plural)).
lf(attribute('noun(227283-1-3)',isa('noun(227283-1-a165)'))).
lf(inst('noun(227283-1-3)','Samson')).
lf(inst('noun(227283-1-a165)',decoy)).
::::::::::::::
227284
::::::::::::::

lf(time('noun(227284-1-5)',preflight)).
lf(inst('noun(227284-1-5)','F/A-18A')).
lf(attribute('noun(227284-2-2)','11')).
lf(location('noun(227284-2-2)',on('noun(227284-1-5)'))).
lf(inst('noun(227284-2-2)','ITER')).
lf(theme('pastpart(227284-1-1)',obj('noun(227284-1-3)'))).
lf(location('pastpart(227284-1-1)',on('noun(227284-1-5)'))).
lf(location('pastpart(227284-1-1)',on('noun(227284-2-2)'))).
lf(activity('pastpart(227284-1-1)',assemble)).
lf(quantity('noun(227284-1-3)',plural)).
lf(attribute('noun(227284-1-3)',isa('noun(227284-1-a167)'))).
lf(inst('noun(227284-1-3)','Samson')).
lf(inst('noun(227284-1-a167)',decoy)).
::::::::::::::
227285
::::::::::::::

lf(time('noun(227285-1-5)',preflight)).
lf(inst('noun(227285-1-5)','F/A-18A')).
lf(theme('noun(227285-2-1)',obj('noun(227285-1-3)'))).
lf(inst('noun(227285-2-1)',view)).
lf(theme('pastpart(227285-1-1)',obj('noun(227285-1-3)'))).
lf(location('pastpart(227285-1-1)',on('noun(227285-1-5)'))).
lf(activity('pastpart(227285-1-1)',assemble)).
lf(attribute('noun(227285-1-3)',isa('noun(227285-1-a168)'))).
lf(quantity('noun(227285-1-3)',plural(2))).
lf(inst('noun(227285-1-3)','Samson')).
lf(inst('noun(227285-1-a168)',decoy)).
::::::::::::::
227286
::::::::::::::

lf(theme('noun(227286-2-1)',obj('noun(227286-1-3)'))).
lf(inst('noun(227286-2-1)','front view')).
lf(time('noun(227286-1-5)',preflight)).
lf(inst('noun(227286-1-5)','F/A-18A')).
lf(theme('pastpart(227286-1-1)',obj('noun(227286-1-3)'))).
lf(location('pastpart(227286-1-1)',on('noun(227286-1-5)'))).
lf(activity('pastpart(227286-1-1)',assemble)).
lf(attribute('noun(227286-1-3)',isa('noun(227286-1-a169)'))).
lf(quantity('noun(227286-1-3)',plural(2))).
lf(inst('noun(227286-1-3)','Samson')).
lf(inst('noun(227286-1-a169)',decoy)).
::::::::::::::
```

```
227462
::::::::::::::

lf(inst('noun(227462-2-a171)',pilot)).
lf(attribute('noun(227462-2-3)',inst('noun(227462-2-a170)'))).
lf(attribute('noun(227462-2-3)',inst('noun(227462-2-a171)'))).
lf(location('noun(227462-2-3)',over('noun(227462-2-5)'))).
lf(attribute('noun(227462-2-3)',employee_of('noun(227462-2-4)'))).
lf(inst('noun(227462-2-3)','Gallinetti')).
lf(inst('noun(227462-2-a170)','Major')).
lf(attribute('noun(227462-1-4)',has_part('noun(227462-1-5)'))).
lf(attribute('noun(227462-1-4)',has_part('noun(227462-1-6)'))).
lf(attribute('noun(227462-1-4)',inst('noun(227462-1-3)'))).
lf(location('noun(227462-1-4)',in('noun(227462-1-a172)'))).
lf(inst('noun(227462-1-4)','F/A-18A')).
lf(attribute('noun(227462-1-3)','161366')).
lf(inst('noun(227462-1-3)',bureau_no)).
lf(theme('noun(227462-1-1)',obj('noun(227462-1-4)'))).
lf(inst('noun(227462-1-1)','air-to-air view')).
lf(location('noun(227462-2-6)',on('noun(227462-2-7)'))).
lf(attribute('noun(227462-2-6)',part_of('noun(227462-2-5)'))).
lf(inst('noun(227462-2-6)',snow)).
lf(inst('noun(227462-2-5)','Olancha Peak')).
lf(inst('noun(227462-2-4)','USMC')).
lf(inst('noun(227462-1-a172)',air)).
lf(inst('noun(227462-2-7)',land)).
lf(attribute('noun(227462-1-6)','1')).
lf(inst('noun(227462-1-6)',tail)).
lf(attribute('noun(227462-1-5)','1')).
lf(inst('noun(227462-1-5)',nose)).
::::::::::::::
228544
::::::::::::::

lf(inst('noun(228544-2-a173)','Kennedy')).
lf(theme('noun(228544-2-1)',obj('noun(228544-1-2)'))).
lf(quantity('noun(228544-2-1)',full)).
lf(theme('noun(228544-2-1)',obj('noun(228544-2-3)'))).
lf(inst('noun(228544-2-1)','side view')).
lf(inst('noun(228544-1-5)','Michelson Laboratory')).
lf(inst('noun(228544-2-5)',rear)).
lf(location('noun(228544-1-2)',before('noun(228544-1-5)'))).
lf(location('noun(228544-1-2)',on('noun(228544-1-4)'))).
lf(inst('noun(228544-1-2)','ACIMD')).
lf(inst('noun(228544-2-4)',rock)).
lf(attribute('noun(228544-2-3)',inst('noun(228544-2-a173)'))).
lf(location('noun(228544-2-3)',at('noun(228544-2-5)'))).
lf(location('noun(228544-2-3)',on('noun(228544-2-4)'))).
lf(inst('noun(228544-2-3)',plaque)).
lf(inst('noun(228544-1-4)',stand)).
::::::::::::::
228545
::::::::::::::

lf(inst('noun(228545-2-a174)','Kennedy')).
lf(theme('noun(228545-2-1)',obj('noun(228545-1-2)'))).
lf(quantity('noun(228545-2-1)','1/4')).
lf(theme('noun(228545-2-1)',obj('noun(228545-2-3)'))).
lf(inst('noun(228545-2-1)','front view')).
lf(inst('noun(228545-1-5)','Michelson Laboratory')).
lf(inst('noun(228545-2-5)',rear)).
```

```
lf(location('noun(228545-1-2)',before('noun(228545-1-5)'))).
lf(location('noun(228545-1-2)',on('noun(228545-1-4)'))).
lf(inst('noun(228545-1-2)','ACIMD')).
lf(inst('noun(228545-2-4)',rock)).
lf(attribute('noun(228545-2-3)',inst('noun(228545-2-a174)'))).
lf(location('noun(228545-2-3)',at('noun(228545-2-5)'))).
lf(location('noun(228545-2-3)',on('noun(228545-2-4)'))).
lf(inst('noun(228545-2-3)',plaque)).
lf(inst('noun(228545-1-4)',stand)).
::::::::::::::
228546
::::::::::::::

lf(inst('noun(228546-2-a175)','Kennedy')).
lf(theme('noun(228546-2-1)',obj('noun(228546-1-2)'))).
lf(theme('noun(228546-2-1)',obj('noun(228546-2-3)'))).
lf(inst('noun(228546-2-1)','side view')).
lf(inst('noun(228546-1-5)','Michelson Laboratory')).
lf(inst('noun(228546-2-5)',rear)).
lf(location('noun(228546-1-2)',before('noun(228546-1-5)'))).
lf(location('noun(228546-1-2)',on('noun(228546-1-4)'))).
lf(inst('noun(228546-1-2)','ACIMD')).
lf(inst('noun(228546-2-4)',rock)).
lf(attribute('noun(228546-2-3)',inst('noun(228546-2-a175)'))).
lf(location('noun(228546-2-3)',at('noun(228546-2-5)'))).
lf(location('noun(228546-2-3)',on('noun(228546-2-4)'))).
lf(inst('noun(228546-2-3)',plaque)).
lf(inst('noun(228546-1-4)',stand)).
::::::::::::::
228795
::::::::::::::

lf(inst('noun(228795-1-a10)','Phoenix')).
lf(attribute('noun(228795-1-4)',inst('noun(228795-1-5)'))).
lf(inst('noun(228795-1-4)','QF-4B')).
lf(attribute('noun(228795-2-1)',2)).
lf(agent('noun(228795-2-1)',obj('noun(228795-1-4)'))).
lf(activity('noun(228795-2-1)',fly)).
lf(time('noun(228795-3-1)',after('pastpart(228795-3-1)'))).
lf(theme('noun(228795-3-1)',obj('noun(228795-1-4)'))).
lf(inst('noun(228795-3-1)',view)).
lf(attribute('noun(228795-1-5)','1')).
lf(inst('noun(228795-1-5)',number)).
lf(attribute('noun(228795-4-1)',large)).
lf(source('noun(228795-4-1)',from('noun(228795-1-4)'))).
lf(inst('noun(228795-4-1)',plume)).
lf(theme('pastpart(228795-3-1)',obj('noun(228795-1-4)'))).
lf(agent('pastpart(228795-3-1)',obj('noun(228795-1-a10)'))).
lf(activity('pastpart(228795-3-1)',hit)).
::::::::::::::
228796
::::::::::::::

lf(inst('noun(228796-1-a11)','Phoenix')).
lf(attribute('noun(228796-1-4)',inst('noun(228796-1-5)'))).
lf(time('noun(228796-1-4)',before('noun(228796-1-3)'))).
lf(inst('noun(228796-1-4)','QF-4B')).
lf(attribute('noun(228796-2-1)',2)).
lf(agent('noun(228796-2-1)',obj('noun(228796-1-4)'))).
lf(activity('noun(228796-2-1)',fly)).
lf(theme('noun(228796-3-1)',obj('noun(228796-1-4)'))).
```

```
lf(inst('noun(228796-3-1)',view)).
lf(attribute('noun(228796-1-5)','1')).
lf(inst('noun(228796-1-5)',number)).
lf(agent('noun(228796-1-3)',obj('noun(228796-1-a11)'))).
lf(theme('noun(228796-1-3)',obj('noun(228796-1-4)'))).
lf(theme('noun(228796-1-3)',obj('noun(228796-1-a11)'))).
lf(activity('noun(228796-1-3)',hit)).
::::::::::::::
228797
::::::::::::::

lf(inst('noun(228797-1-a12)','Phoenix')).
lf(attribute('noun(228797-1-4)',inst('noun(228797-1-5)'))).
lf(inst('noun(228797-1-4)','QF-4B')).
lf(attribute('noun(228797-2-1)',2)).
lf(agent('noun(228797-2-1)',obj('noun(228797-1-4)'))).
lf(activity('noun(228797-2-1)',fly)).
lf(theme('noun(228797-3-5)',obj('noun(228797-1-a12)'))).
lf(state('noun(228797-3-5)',shape)).
lf(inst('noun(228797-3-5)',explosion)).
lf(theme('noun(228797-3-1)',obj('noun(228797-1-3)'))).
lf(inst('noun(228797-3-1)',view)).
lf(attribute('noun(228797-1-5)','1')).
lf(inst('noun(228797-1-5)',number)).
lf(agent('noun(228797-1-3)',obj('noun(228797-1-a12)'))).
lf(theme('noun(228797-1-3)',obj('noun(228797-1-4)'))).
lf(accompaniment('noun(228797-1-3)',inst('noun(228797-3-5)'))).
lf(activity('noun(228797-1-3)',hit)).
::::::::::::::
230834
::::::::::::::

lf(attribute('noun(230834-1-3)',has_part('noun(230834-1-4)'))).
lf(attribute('noun(230834-1-3)',inst('noun(230834-1-5)'))).
lf(attribute('noun(230834-1-3)',inst('noun(230834-1-2)'))).
lf(inst('noun(230834-1-3)','F/A-18A')).
lf(attribute('noun(230834-1-2)','161720')).
lf(inst('noun(230834-1-2)',bureau_no)).
lf(location('noun(230834-2-3)',on('noun(230834-1-10)'))).
lf(inst('noun(230834-2-3)',center)).
lf(quantity('noun(230834-2-1)',closeup)).
lf(theme('noun(230834-2-1)',obj('noun(230834-1-3)'))).
lf(theme('noun(230834-2-1)',obj('noun(230834-2-3)'))).
lf(inst('noun(230834-2-1)',view)).
lf(location('noun(230834-1-8)',over('noun(230834-1-10)'))).
lf(location('noun(230834-1-8)',on('noun(230834-1-3)'))).
lf(inst('noun(230834-1-8)','Harm')).
lf(location('noun(230834-1-5)',on('noun(230834-1-6)'))).
lf(inst('noun(230834-1-5)','flying eagle')).
lf(attribute('noun(230834-1-4)','102')).
lf(inst('noun(230834-1-4)',side)).
lf(inst('noun(230834-1-10)',sea)).
lf(inst('noun(230834-1-6)',tail)).
::::::::::::::
231373
::::::::::::::

lf(inst('noun(231373-2-4)','MK-91')).
lf(location('noun(231373-1-6)',over('noun(231373-1-9)'))).
lf(attribute('noun(231373-1-6)',has_part('noun(231373-1-7)'))).
lf(attribute('noun(231373-1-6)',inst('noun(231373-1-5)'))).
```

```
lf(time('noun(231373-1-6)',before('pastpart(231373-1-1)'))).
lf(location('noun(231373-1-6)',in('noun(231373-1-a180)'))).
lf(inst('noun(231373-1-6)','F/A-18A')).
lf(attribute('noun(231373-1-5)','161720')).
lf(inst('noun(231373-1-5)',bureau_no)).
lf(theme('noun(231373-1-1)',obj('noun(231373-1-3)'))).
lf(attribute('noun(231373-1-1)',excellent)).
lf(theme('noun(231373-1-1)',obj('noun(231373-1-6)'))).
lf(inst('noun(231373-1-1)','air-to-air view')).
lf(theme('pastpart(231373-2-1)',obj('noun(231373-1-3)'))).
lf(theme('pastpart(231373-2-1)',obj('noun(231373-2-2)'))).
lf(theme('pastpart(231373-2-1)',obj('noun(231373-2-4)'))).
lf(location('pastpart(231373-2-1)',on('noun(231373-1-6)'))).
lf(activity('pastpart(231373-2-1)',assemble)).
lf(inst('noun(231373-2-2)',pod)).
lf(location('noun(231373-1-3)',in('noun(231373-1-a180)'))).
lf(inst('noun(231373-1-3)','Harm')).
lf(inst('noun(231373-1-a180)',air)).
lf(inst('noun(231373-1-a179)','NWC')).
lf(theme('pastpart(231373-1-1)',obj('noun(231373-1-3)'))).
lf(source('pastpart(231373-1-1)',from('noun(231373-1-6)'))).
lf(activity('pastpart(231373-1-1)',launch)).
lf(quantity('noun(231373-1-9)',plural)).
lf(attribute('noun(231373-1-9)',inst('noun(231373-1-a179)'))).
lf(inst('noun(231373-1-9)',range)).
lf(attribute('noun(231373-1-7)','102')).
lf(inst('noun(231373-1-7)',nose)).
:::::::::::::
231374
:::::::::::::

lf(agent('noun(231374-5-3)',obj('noun(231374-1-3)'))).
lf(inst('noun(231374-5-3)',exhaust)).
lf(inst('noun(231374-2-4)','MK-91')).
lf(quantity('noun(231374-3-1)',full)).
lf(attribute('noun(231374-3-1)',excellent)).
lf(theme('noun(231374-3-1)',obj('noun(231374-1-6)'))).
lf(inst('noun(231374-3-1)','side view')).
lf(location('noun(231374-1-6)',over('noun(231374-1-9)'))).
lf(attribute('noun(231374-1-6)',has_part('noun(231374-1-7)'))).
lf(attribute('noun(231374-1-6)',inst('noun(231374-1-5)'))).
lf(time('noun(231374-1-6)',after('pastpart(231374-1-1)'))).
lf(inst('noun(231374-1-6)','F/A-18A')).
lf(attribute('noun(231374-1-5)','161720')).
lf(inst('noun(231374-1-5)',bureau_no)).
lf(theme('noun(231374-1-1)',obj('noun(231374-1-3)'))).
lf(inst('noun(231374-1-1)','air-to-air view')).
lf(theme('pastpart(231374-2-1)',obj('noun(231374-1-3)'))).
lf(theme('pastpart(231374-2-1)',obj('noun(231374-2-2)'))).
lf(theme('pastpart(231374-2-1)',obj('noun(231374-2-4)'))).
lf(location('pastpart(231374-2-1)',before('noun(231374-1-6)'))).
lf(location('pastpart(231374-2-1)',on('noun(231374-1-6)'))).
lf(activity('pastpart(231374-2-1)',assemble)).
lf(inst('noun(231374-2-2)',pod)).
lf(location('noun(231374-1-3)',in('noun(231374-1-a182)'))).
lf(inst('noun(231374-1-3)','Harm')).
lf(inst('noun(231374-1-a182)',air)).
lf(attribute('noun(231374-5-1)',diamond)).
lf(attribute('noun(231374-5-1)',excellent)).
lf(location('noun(231374-5-1)',in('noun(231374-5-3)'))).
lf(inst('noun(231374-5-1)',pattern)).
```

```
lf(inst('noun(231374-1-a181)','NWC')).
lf(theme('pastpart(231374-1-1)',obj('noun(231374-1-3)'))).
lf(source('pastpart(231374-1-1)',from('noun(231374-1-6)'))).
lf(activity('pastpart(231374-1-1)',launch)).
lf(quantity('noun(231374-1-9)',plural)).
lf(attribute('noun(231374-1-9)',inst('noun(231374-1-a181)'))).
lf(inst('noun(231374-1-9)',range)).
lf(attribute('noun(231374-1-7)','102')).
lf(inst('noun(231374-1-7)',nose)).
::::::::::::::
231375
::::::::::::::

lf(agent('noun(231375-4-3)',obj('noun(231375-1-3)'))).
lf(inst('noun(231375-4-3)',exhaust)).
lf(inst('noun(231375-2-4)','MK-91')).
lf(attribute('noun(231375-3-1)',excellent)).
lf(theme('noun(231375-3-1)',obj('noun(231375-1-6)'))).
lf(inst('noun(231375-3-1)','side view')).
lf(location('noun(231375-1-6)',over('noun(231375-1-9)'))).
lf(attribute('noun(231375-1-6)',has_part('noun(231375-1-7)'))).
lf(attribute('noun(231375-1-6)',inst('noun(231375-1-5)'))).
lf(time('noun(231375-1-6)',at('pastpart(231375-1-1)'))).
lf(inst('noun(231375-1-6)','F/A-18A')).
lf(attribute('noun(231375-1-5)','161720')).
lf(inst('noun(231375-1-5)',bureau_no)).
lf(theme('noun(231375-1-1)',obj('noun(231375-1-3)'))).
lf(inst('noun(231375-1-1)','air-to-air view')).
lf(theme('pastpart(231375-2-1)',obj('noun(231375-1-3)'))).
lf(theme('pastpart(231375-2-1)',obj('noun(231375-2-2)'))).
lf(theme('pastpart(231375-2-1)',obj('noun(231375-2-4)'))).
lf(location('pastpart(231375-2-1)',on('noun(231375-1-6)'))).
lf(activity('pastpart(231375-2-1)',assemble)).
lf(inst('noun(231375-2-2)',pod)).
lf(location('noun(231375-1-3)',in('noun(231375-1-a184)'))).
lf(inst('noun(231375-1-3)','Harm')).
lf(inst('noun(231375-1-a184)',air)).
lf(attribute('noun(231375-4-1)',diamond)).
lf(attribute('noun(231375-4-1)',excellent)).
lf(location('noun(231375-4-1)',in('noun(231375-4-3)'))).
lf(inst('noun(231375-4-1)',pattern)).
lf(inst('noun(231375-1-a183)','NWC')).
lf(theme('pastpart(231375-1-1)',obj('noun(231375-1-3)'))).
lf(source('pastpart(231375-1-1)',from('noun(231375-1-6)'))).
lf(activity('pastpart(231375-1-1)',launch)).
lf(quantity('noun(231375-1-9)',plural)).
lf(attribute('noun(231375-1-9)',inst('noun(231375-1-a183)'))).
lf(inst('noun(231375-1-9)',range)).
lf(attribute('noun(231375-1-7)','102')).
lf(inst('noun(231375-1-7)',nose)).
::::::::::::::
232744
::::::::::::::

lf(location('noun(232744-1-9)',on('noun(232744-1-4)'))).
lf(inst('noun(232744-1-9)','B57-VFA-82A')).
lf(theme('noun(232744-2-1)',obj('noun(232744-1-4)'))).
lf(inst('noun(232744-2-1)','side view')).
lf(attribute('noun(232744-1-4)',has_part('noun(232744-1-5)'))).
lf(attribute('noun(232744-1-4)',inst('noun(232744-1-3)'))).
lf(location('noun(232744-1-4)',over('noun(232744-2-5)'))).
```

```
lf(location('noun(232744-1-4)',in('noun(232744-1-a187)'))).
lf(inst('noun(232744-1-4)','F/A-18A')).
lf(attribute('noun(232744-1-3)','161720')).
lf(inst('noun(232744-1-3)',bureau_no)).
lf(theme('noun(232744-1-1)',obj('noun(232744-1-4)'))).
lf(inst('noun(232744-1-1)','air-to-air view')).
lf(location('noun(232744-1-7)',on('noun(232744-1-4)'))).
lf(inst('noun(232744-1-7)','Harm')).
lf(inst('noun(232744-1-a187)',air)).
lf(inst('noun(232744-2-a186)','NWC')).
lf(inst('noun(232744-2-a185)',range)).
lf(quantity('noun(232744-2-5)',plural)).
lf(attribute('noun(232744-2-5)',inst('noun(232744-2-a185)'))).
lf(attribute('noun(232744-2-5)',inst('noun(232744-2-a186)'))).
lf(inst('noun(232744-2-5)',mountain)).
lf(attribute('noun(232744-1-5)','102')).
lf(inst('noun(232744-1-5)',nose)).
::::::::::::::
232745
::::::::::::::

lf(location('noun(232745-1-9)',on('noun(232745-1-4)'))).
lf(inst('noun(232745-1-9)','B57-VFA-82A')).
lf(time('noun(232745-2-1)',during('prespart(232745-2-1)'))).
lf(theme('noun(232745-2-1)',obj('noun(232745-1-4)'))).
lf(inst('noun(232745-2-1)','side view')).
lf(attribute('noun(232745-1-4)',has_part('noun(232745-1-5)'))).
lf(attribute('noun(232745-1-4)',inst('noun(232745-1-3)'))).
lf(location('noun(232745-1-4)',in('noun(232745-1-a189)'))).
lf(inst('noun(232745-1-4)','F/A-18A')).
lf(attribute('noun(232745-1-3)','161720')).
lf(inst('noun(232745-1-3)',bureau_no)).
lf(theme('noun(232745-1-1)',obj('noun(232745-1-4)'))).
lf(theme('noun(232745-1-1)',obj('noun(232745-3-1)'))).
lf(attribute('noun(232745-1-1)',inst('noun(232745-3-2)'))).
lf(inst('noun(232745-1-1)','air-to-air view')).
lf(inst('noun(232745-3-2)',center)).
lf(location('noun(232745-3-1)',in('noun(232745-3-2)'))).
lf(location('noun(232745-3-1)',in('noun(232745-1-a189)'))).
lf(inst('noun(232745-3-1)',wash)).
lf(location('noun(232745-1-7)',on('noun(232745-1-4)'))).
lf(inst('noun(232745-1-7)','Harm')).
lf(inst('noun(232745-1-a189)',air)).
lf(inst('noun(232745-2-a188)','NWC')).
lf(agent('prespart(232745-2-1)',obj('noun(232745-1-4)'))).
lf(location('prespart(232745-2-1)',over('noun(232745-2-4)'))).
lf(activity('prespart(232745-2-1)',ptrans)).
lf(quantity('noun(232745-2-4)',plural)).
lf(attribute('noun(232745-2-4)',inst('noun(232745-2-a188)'))).
lf(inst('noun(232745-2-4)',range)).
lf(attribute('noun(232745-1-5)','102')).
lf(inst('noun(232745-1-5)',nose)).
::::::::::::::
232747
::::::::::::::

lf(location('noun(232747-1-9)',on('noun(232747-1-4)'))).
lf(inst('noun(232747-1-9)','B57-VFA-82A')).
lf(time('noun(232747-2-1)',during('prespart(232747-2-1)'))).
lf(theme('noun(232747-2-1)',obj('noun(232747-1-4)'))).
lf(inst('noun(232747-2-1)','side view')).
```

```
lf(attribute('noun(232747-1-4)',has_part('noun(232747-1-5)'))).
lf(attribute('noun(232747-1-4)',inst('noun(232747-1-3)'))).
lf(location('noun(232747-1-4)',in('noun(232747-1-a191)'))).
lf(inst('noun(232747-1-4)','F/A-18A')).
lf(attribute('noun(232747-1-3)','161720')).
lf(inst('noun(232747-1-3)',bureau_no)).
lf(theme('noun(232747-1-1)',obj('noun(232747-1-4)'))).
lf(inst('noun(232747-1-1)','air-to-air view')).
lf(location('noun(232747-1-7)',on('noun(232747-1-4)'))).
lf(inst('noun(232747-1-7)','Harm')).
lf(inst('noun(232747-1-a191)',air)).
lf(quantity('noun(232747-3-1)',full)).
lf(attribute('noun(232747-3-1)',part_of('noun(232747-1-4)'))).
lf(inst('noun(232747-3-1)',frame)).
lf(inst('noun(232747-2-a190)','NWC')).
lf(agent('prespart(232747-2-1)',obj('noun(232747-1-4)'))).
lf(location('prespart(232747-2-1)',over('noun(232747-2-4)'))).
lf(activity('prespart(232747-2-1)',ptrans)).
lf(quantity('noun(232747-2-4)',plural)).
lf(attribute('noun(232747-2-4)',inst('noun(232747-2-a190)'))).
lf(inst('noun(232747-2-4)',range)).
lf(attribute('noun(232747-1-5)','102')).
lf(inst('noun(232747-1-5)',nose)).
::::::::::::::
234050
::::::::::::::

lf(inst('noun(234050-2-a193)','J-52')).
lf(inst('noun(234050-1-1)','Weapons Survivability Laboratory')).
lf(theme('noun(234050-2-4)',obj('noun(234050-2-a192)'))).
lf(inst('noun(234050-2-4)',ingestion)).
lf(attribute('noun(234050-2-a192)',consumed_by('noun(234050-2-a193)'))).
lf(inst('noun(234050-2-a192)',fuel)).
lf(source('noun(234050-2-1)',from('noun(234050-2-4)'))).
lf(inst('noun(234050-2-1)',fireball)).
::::::::::::::
234064
::::::::::::::

lf(location('noun(234064-1-2)',on('noun(234064-1-4)'))).
lf(inst('noun(234064-1-2)','GBU-22B')).
lf(theme('noun(234064-2-1)',obj('noun(234064-1-2)'))).
lf(attribute('noun(234064-2-1)',left)).
lf(inst('noun(234064-2-1)','side view')).
lf(theme('pres(234064-3-1)',obj('noun(234064-3-1)'))).
lf(activity('pres(234064-3-1)',open_act)).
lf(quantity('noun(234064-3-1)',plural)).
lf(inst('noun(234064-3-1)',fin)).
lf(inst('noun(234064-1-4)',stand)).
::::::::::::::
234756
::::::::::::::

lf(quantity('noun(234756-3-1)',full)).
lf(theme('noun(234756-3-1)',obj('noun(234756-1-4)'))).
lf(inst('noun(234756-3-1)','side view')).
lf(attribute('noun(234756-1-4)',inst('noun(234756-1-3)'))).
lf(location('noun(234756-1-4)',under('noun(234756-4-2)'))).
lf(location('noun(234756-1-4)',in('noun(234756-1-a194)'))).
lf(inst('noun(234756-1-4)','QF-4H-1')).
lf(attribute('noun(234756-1-3)','149452')).
```

```
lf(inst('noun(234756-1-3)',bureau_no)).
lf(theme('noun(234756-1-1)',obj('noun(234756-1-4)'))).
lf(quantity('noun(234756-1-1)',full)).
lf(theme('noun(234756-1-1)',obj('noun(234756-4-2)'))).
lf(inst('noun(234756-1-1)','air-to-air view')).
lf(location('noun(234756-4-2)',in('noun(234756-1-a194)'))).
lf(inst('noun(234756-4-2)','Armitage Field')).
lf(location('noun(234756-2-3)',on('noun(234756-2-4)'))).
lf(location('noun(234756-2-3)',under('noun(234756-2-6)'))).
lf(location('noun(234756-2-3)',on('noun(234756-2-8)'))).
lf(location('noun(234756-2-3)',on('noun(234756-2-10)'))).
lf(attribute('noun(234756-2-3)',part_of('noun(234756-2-2)'))).
lf(inst('noun(234756-2-3)',red)).
lf(inst('noun(234756-2-2)',silver)).
lf(inst('noun(234756-1-a194)',air)).
lf(theme('pastpart(234756-2-1)',obj('noun(234756-2-2)'))).
lf(agent('pastpart(234756-2-1)',obj('noun(234756-1-4)'))).
lf(activity('pastpart(234756-2-1)',color)).
lf(inst('noun(234756-2-10)',fin)).
lf(inst('noun(234756-2-8)',tail)).
lf(quantity('noun(234756-2-6)',plural)).
lf(inst('noun(234756-2-6)',wing)).
lf(inst('noun(234756-2-4)',nose)).
::::::::::::::
237492
::::::::::::::

lf(inst('noun(237492-2-a195)','AV-8B')).
lf(attribute('noun(237492-2-2)',part_of('noun(237492-2-a195)'))).
lf(inst('noun(237492-2-2)','DVT-2R')).
lf(inst('noun(237492-5-2)','drogue parachute')).
lf(attribute('coordinate(237492-3-1)','1900 '' N x 21 '' W')).
lf(inst('coordinate(237492-3-1)',coordinate)).
lf(theme('noun(237492-3-2)',obj('noun(237492-3-a196)'))).
lf(location('noun(237492-3-2)',at('coordinate(237492-3-1)'))).
lf(quantity('noun(237492-3-2)',kts('225'))).
lf(inst('noun(237492-3-2)',ejection)).
lf(agent('prespart(237492-6-1)',obj('noun(237492-6-2)'))).
lf(activity('prespart(237492-6-1)',launch)).
lf(quantity('noun(237492-6-2)',plural)).
lf(attribute('noun(237492-6-2)',part_of('noun(237492-3-a196)'))).
lf(inst('noun(237492-6-2)',rocket)).
lf(theme('pastpart(237492-5-1)',obj('noun(237492-5-2)'))).
lf(attribute('pastpart(237492-5-1)',no)).
lf(activity('pastpart(237492-5-1)',open_act)).
lf(theme('be(237492-5-1)',obj('noun(237492-5-2)'))).
lf(location('be(237492-5-1)',above('noun(237492-4-1)'))).
lf(activity('be(237492-5-1)',be)).
lf(agent('prespart(237492-4-1)',obj('noun(237492-4-1)'))).
lf(activity('prespart(237492-4-1)',expel)).
lf(attribute('noun(237492-4-1)',rear)).
lf(inst('noun(237492-4-1)',dummy)).
lf(inst('noun(237492-3-a196)',seat)).
lf(attribute('noun(237492-1-1)','85221')).
lf(inst('noun(237492-1-1)','test plan')).
::::::::::::::
238225
::::::::::::::

lf(theme('noun(238225-2-1)',obj('noun(238225-1-4)'))).
lf(inst('noun(238225-2-1)','front view')).
```

```
lf(attribute('noun(238225-1-4)',inst('noun(238225-1-2)'))).
lf(inst('noun(238225-1-4)','A-3B')).
lf(location('noun(238225-1-7)',in('noun(238225-1-8)'))).
lf(attribute('noun(238225-1-7)',part_of('noun(238225-1-4)'))).
lf(inst('noun(238225-1-7)','EX-62')).
lf(theme('noun(238225-1-9)',obj('noun(238225-1-5)'))).
lf(inst('noun(238225-1-9)','CFT')).
lf(location('noun(238225-1-5)',on('noun(238225-1-4)'))).
lf(inst('noun(238225-1-5)','Standard Arm II-N')).
lf(attribute('noun(238225-1-2)','142630')).
lf(inst('noun(238225-1-2)',bureau_no)).
lf(theme('pastpart(238225-2-1)',obj('noun(238225-1-7)'))).
lf(activity('pastpart(238225-2-1)',assemble)).
lf(reason('noun(238225-1-8)',for('noun(238225-1-9)'))).
lf(inst('noun(238225-1-8)',nose)).
::::::::::::::
238226
::::::::::::::

lf(quantity('noun(238226-2-1)',full)).
lf(theme('noun(238226-2-1)',obj('noun(238226-1-4)'))).
lf(inst('noun(238226-2-1)','frame view')).
lf(attribute('noun(238226-1-4)',inst('noun(238226-1-3)'))).
lf(location('noun(238226-1-4)',over('noun(238226-2-3)'))).
lf(location('noun(238226-1-4)',in('noun(238226-1-a197)'))).
lf(inst('noun(238226-1-4)','A-3B')).
lf(location('noun(238226-1-7)',in('noun(238226-1-8)'))).
lf(attribute('noun(238226-1-7)',part_of('noun(238226-1-4)'))).
lf(inst('noun(238226-1-7)','EX-62')).
lf(theme('noun(238226-1-9)',obj('noun(238226-1-5)'))).
lf(inst('noun(238226-1-9)','CFT')).
lf(location('noun(238226-1-5)',on('noun(238226-1-4)'))).
lf(inst('noun(238226-1-5)','Standard Arm II-N')).
lf(attribute('noun(238226-1-3)','142630')).
lf(inst('noun(238226-1-3)',bureau_no)).
lf(theme('noun(238226-1-1)',obj('noun(238226-1-4)'))).
lf(inst('noun(238226-1-1)','air-to-air view')).
lf(inst('noun(238226-1-a197)',air)).
lf(inst('noun(238226-2-4)','NWC')).
lf(attribute('noun(238226-2-3)',north)).
lf(attribute('noun(238226-2-3)',part_of('noun(238226-2-4)'))).
lf(inst('noun(238226-2-3)',range)).
lf(reason('noun(238226-1-8)',for('noun(238226-1-9)'))).
lf(inst('noun(238226-1-8)',nose)).
::::::::::::::
239091
::::::::::::::

lf(theme('noun(239091-2-4)',obj('noun(239091-2-3)'))).
lf(inst('noun(239091-2-4)','overall view')).
lf(inst('noun(239091-2-a200)',validation)).
lf(inst('noun(239091-1-a198)','WSSA')).
lf(attribute('noun(239091-1-2)',part_of('noun(239091-1-a198)'))).
lf(inst('noun(239091-1-2)','F/A-18A Validation Laboratory')).
lf(inst('noun(239091-2-a199)',cockpit)).
lf(attribute('noun(239091-2-3)',isa('noun(239091-2-a199)'))).
lf(attribute('noun(239091-2-3)',inst('noun(239091-2-a200)'))).
lf(inst('noun(239091-2-3)',station)).
::::::::::::::
239092
::::::::::::::
```

```
lf(inst('noun(239092-2-a204)',center)).
lf(inst('noun(239092-2-a203)',validation)).
lf(attribute('noun(239092-2-2)',inst('noun(239092-2-a204)'))).
lf(attribute('noun(239092-2-2)',part_of('noun(239092-2-5)'))).
lf(inst('noun(239092-2-2)',console)).
lf(inst('noun(239092-1-a201)','WSSA')).
lf(attribute('noun(239092-1-2)',part_of('noun(239092-1-a201)'))).
lf(inst('noun(239092-1-2)','F/A-18A Validation Laboratory')).
lf(inst('noun(239092-2-a202)',cockpit)).
lf(attribute('noun(239092-2-5)',isa('noun(239092-2-a202)'))).
lf(attribute('noun(239092-2-5)',inst('noun(239092-2-a203)'))).
lf(inst('noun(239092-2-5)',station)).
:::::::::::::
239093
:::::::::::::

lf(inst('noun(239093-2-a207)','John')).
lf(attribute('noun(239093-2-7)',inst('noun(239093-2-a206)'))).
lf(location('noun(239093-2-7)',on('dir(239093-2-2)'))).
lf(attribute('noun(239093-2-7)',citizen_of('noun(239093-2-8)'))).
lf(inst('noun(239093-2-7)','Germain')).
lf(attribute('noun(239093-2-5)',inst('noun(239093-2-a207)'))).
lf(location('noun(239093-2-5)',on('dir(239093-2-1)'))).
lf(inst('noun(239093-2-5)','Hessler')).
lf(inst('noun(239093-2-a206)','Captain')).
lf(inst('noun(239093-2-a209)',validation)).
lf(inst('dir(239093-2-2)',right)).
lf(inst('noun(239093-2-8)','Canada')).
lf(inst('dir(239093-2-1)',left)).
lf(inst('noun(239093-1-a205)','WSSA')).
lf(attribute('noun(239093-1-2)',part_of('noun(239093-1-a205)'))).
lf(inst('noun(239093-1-2)','F/A-18A Validation Laboratory')).
lf(inst('noun(239093-2-a208)',cockpit)).
lf(attribute('noun(239093-2-3)',isa('noun(239093-2-a208)'))).
lf(attribute('noun(239093-2-3)',inst('noun(239093-2-a209)'))).
lf(attribute('noun(239093-2-3)',next_to('noun(239093-2-5)'))).
lf(attribute('noun(239093-2-3)',next_to('noun(239093-2-7)'))).
lf(inst('noun(239093-2-3)',station)).
:::::::::::::
239094
:::::::::::::

lf(inst('noun(239094-2-a211)','Stephen')).
lf(attribute('noun(239094-2-6)',inst('noun(239094-2-a211)'))).
lf(attribute('noun(239094-2-6)',inst('noun(239094-2-a212)'))).
lf(attribute('noun(239094-2-6)',citizen_of('noun(239094-2-7)'))).
lf(inst('noun(239094-2-6)','Germain')).
lf(inst('noun(239094-2-a212)','Captain')).
lf(inst('noun(239094-2-a214)',validation)).
lf(inst('noun(239094-2-7)','Canada')).
lf(inst('noun(239094-1-a210)','WSSA')).
lf(attribute('noun(239094-1-2)',part_of('noun(239094-1-a210)'))).
lf(inst('noun(239094-1-2)','F/A-18A Validation Laboratory')).
lf(inst('noun(239094-2-a213)',cockpit)).
lf(attribute('noun(239094-2-3)',isa('noun(239094-2-a213)'))).
lf(attribute('noun(239094-2-3)',inst('noun(239094-2-a214)'))).
lf(attribute('noun(239094-2-3)',next_to('noun(239094-2-6)'))).
lf(inst('noun(239094-2-3)',station)).
:::::::::::::
239095
```

```
::::::::::::::

lf(inst('noun(239095-2-a216)','John')).
lf(attribute('noun(239095-2-5)',inst('noun(239095-2-a216)'))).
lf(inst('noun(239095-2-5)','Hessler')).
lf(inst('noun(239095-2-a218)',validation)).
lf(inst('noun(239095-1-a215)','WSSA')).
lf(attribute('noun(239095-1-2)',part_of('noun(239095-1-a215)'))).
lf(inst('noun(239095-1-2)','F/A-18A Validation Laboratory')).
lf(inst('noun(239095-2-a217)',cockpit)).
lf(attribute('noun(239095-2-3)',isa('noun(239095-2-a217)'))).
lf(attribute('noun(239095-2-3)',inst('noun(239095-2-a218)'))).
lf(attribute('noun(239095-2-3)',next_to('noun(239095-2-5)'))).
lf(inst('noun(239095-2-3)',station)).
::::::::::::::
239097
::::::::::::::

lf(inst('noun(239097-2-a221)',validation)).
lf(quantity('noun(239097-2-1)',plural)).
lf(attribute('noun(239097-2-1)',avionic)).
lf(attribute('noun(239097-2-1)',part_of('noun(239097-2-4)'))).
lf(inst('noun(239097-2-1)',rack)).
lf(inst('noun(239097-1-a219)','WSSA')).
lf(attribute('noun(239097-1-2)',part_of('noun(239097-1-a219)'))).
lf(inst('noun(239097-1-2)','F/A-18A Validation Laboratory')).
lf(inst('noun(239097-2-a220)',cockpit)).
lf(attribute('noun(239097-2-4)',isa('noun(239097-2-a220)'))).
lf(attribute('noun(239097-2-4)',inst('noun(239097-2-a221)'))).
lf(inst('noun(239097-2-4)',station)).
::::::::::::::
239098
::::::::::::::

lf(inst('noun(239098-2-a224)',validation)).
lf(quantity('noun(239098-2-2)',plural)).
lf(attribute('noun(239098-2-2)',part_of('noun(239098-2-5)'))).
lf(inst('noun(239098-2-2)',computer)).
lf(inst('noun(239098-1-a222)','WSSA')).
lf(attribute('noun(239098-1-2)',part_of('noun(239098-1-a222)'))).
lf(inst('noun(239098-1-2)','F/A-18A Validation Laboratory')).
lf(inst('noun(239098-2-a223)',cockpit)).
lf(attribute('noun(239098-2-5)',isa('noun(239098-2-a223)'))).
lf(attribute('noun(239098-2-5)',inst('noun(239098-2-a224)'))).
lf(inst('noun(239098-2-5)',station)).
::::::::::::::
239126
::::::::::::::

lf(location('noun(239126-2-a225)',on('noun(239126-1-3)'))).
lf(inst('noun(239126-2-a225)','AAR-47')).
lf(attribute('noun(239126-2-5)','142630')).
lf(inst('noun(239126-2-5)',bureau_no)).
lf(attribute('noun(239126-1-3)',inst('noun(239126-2-5)'))).
lf(location('noun(239126-1-3)',in('noun(239126-3-2)'))).
lf(inst('noun(239126-1-3)','A-3B')).
lf(theme('noun(239126-1-1)',obj('noun(239126-1-3)'))).
lf(attribute('noun(239126-1-1)',inst('noun(239126-3-2)'))).
lf(attribute('noun(239126-1-1)',excellent)).
lf(theme('noun(239126-1-1)',obj('noun(239126-2-7)'))).
lf(theme('noun(239126-1-1)',obj('noun(239126-5-3)'))).
```

```
lf(inst('noun(239126-1-1)','air-to-ground view')).
lf(quantity('noun(239126-4-1)',plural(3))).
lf(quantity('noun(239126-4-1)',all)).
lf(inst('noun(239126-4-1)',hangar)).
lf(inst('noun(239126-3-2)',center)).
lf(inst('noun(239126-2-7)','Armitage Field')).
lf(theme('noun(239126-2-3)',obj('noun(239126-2-a225)'))).
lf(location('noun(239126-2-3)',over('noun(239126-2-7)'))).
lf(inst('noun(239126-2-3)',test)).
lf(quantity('noun(239126-5-3)',plural)).
lf(inst('noun(239126-5-3)',mountain)).
lf(agent('prespart(239126-4-1)',obj('noun(239126-4-1)'))).
lf(direction('prespart(239126-4-1)',west)).
lf(activity('prespart(239126-4-1)',observe)).
::::::::::::::
241426
::::::::::::::

lf(inst('noun(241426-2-1)',tank)).
lf(agent('prespart(241426-1-1)',obj('noun(241426-1-2)'))).
lf(destination('prespart(241426-1-1)',obj('noun(241426-1-3)'))).
lf(activity('prespart(241426-1-1)',approach)).
lf(inst('noun(241426-1-2)','Sidearm')).
lf(attribute('noun(241426-1-5)',part_of('noun(241426-1-4)'))).
lf(inst('noun(241426-1-5)',checkerboard)).
lf(theme('be(241426-2-1)',obj('noun(241426-2-1)'))).
lf(theme('be(241426-2-1)',obj('noun(241426-2-2)'))).
lf(location('be(241426-2-1)',before('noun(241426-1-3)'))).
lf(activity('be(241426-2-1)',be)).
lf(inst('noun(241426-2-2)',radar)).
lf(inst('noun(241426-1-4)',pole)).
lf(attribute('noun(241426-1-3)',isa('noun(241426-1-4)'))).
lf(inst('noun(241426-1-3)',target)).
::::::::::::::
241427
::::::::::::::

lf(inst('noun(241427-2-1)',tank)).
lf(inst('noun(241427-1-2)','Sidearm')).
lf(attribute('noun(241427-1-5)',part_of('noun(241427-1-4)'))).
lf(inst('noun(241427-1-5)',checkerboard)).
lf(inst('noun(241427-2-2)',radar)).
lf(agent('prespart(241427-1-1)',obj('noun(241427-1-2)'))).
lf(theme('prespart(241427-1-1)',obj('noun(241427-1-3)'))).
lf(activity('prespart(241427-1-1)',hit)).
lf(inst('noun(241427-1-4)',pole)).
lf(attribute('noun(241427-1-3)',isa('noun(241427-1-4)'))).
lf(inst('noun(241427-1-3)',target)).
::::::::::::::
241452
::::::::::::::

lf(attribute('noun(241452-1-4)',has_part('noun(241452-1-5)'))).
lf(attribute('noun(241452-1-4)',inst('noun(241452-1-6)'))).
lf(inst('noun(241452-1-4)','AV-8B')).
lf(location('noun(241452-1-6)',on('noun(241452-1-7)'))).
lf(inst('noun(241452-1-6)','Marines')).
lf(quantity('noun(241452-1-1)',full)).
lf(theme('noun(241452-1-1)',obj('noun(241452-1-4)'))).
lf(inst('noun(241452-1-1)','side view')).
lf(inst('noun(241452-1-7)',tail)).
```

```
lf(attribute('noun(241452-1-5)','622')).
lf(inst('noun(241452-1-5)',nose)).
::::::::::::::
241950
::::::::::::::

lf(inst('noun(241950-1-5)','AH-1J')).
lf(theme('noun(241950-3-1)',obj('noun(241950-3-2)'))).
lf(theme('noun(241950-3-1)',obj('noun(241950-1-5)'))).
lf(inst('noun(241950-3-1)',view)).
lf(inst('noun(241950-1-a226)','Sidearm')).
lf(theme('noun(241950-1-3)',obj('noun(241950-1-a226)'))).
lf(source('noun(241950-1-3)',from('noun(241950-1-5)'))).
lf(activity('noun(241950-1-3)',launch)).
lf(inst('noun(241950-3-2)',plume)).
lf(theme('be(241950-2-1)',obj('noun(241950-1-a226)'))).
lf(source('be(241950-2-1)',from('noun(241950-1-5)'))).
lf(activity('be(241950-2-1)',be)).
::::::::::::::
242099
::::::::::::::

lf(attribute('noun(242099-2-4)',owned_by('noun(242099-2-a229)'))).
lf(location('noun(242099-2-4)',over('noun(242099-2-5)'))).
lf(inst('noun(242099-2-4)','AH-1W')).
lf(inst('noun(242099-2-5)','G-2 Range')).
lf(inst('noun(242099-1-a228)','AIM-9L')).
lf(inst('noun(242099-2-a229)','VX-5')).
lf(inst('noun(242099-1-a227)','OPNAV')).
lf(attribute('noun(242099-1-4)',inst('noun(242099-1-a227)'))).
lf(theme('noun(242099-1-4)',obj('noun(242099-1-a228)'))).
lf(inst('noun(242099-1-4)',evaluation)).
lf(attribute('noun(242099-4-1)',large)).
lf(location('noun(242099-4-1)',behind('noun(242099-1-a228)'))).
lf(inst('noun(242099-4-1)',plume)).
lf(theme('pastpart(242099-2-1)',obj('noun(242099-1-a228)'))).
lf(source('pastpart(242099-2-1)',from('noun(242099-2-4)'))).
lf(location('pastpart(242099-2-1)',before('noun(242099-2-4)'))).
lf(activity('pastpart(242099-2-1)',launch)).
::::::::::::::
242109
::::::::::::::

lf(location('noun(242109-1-4)',in('noun(242109-1-a231)'))).
lf(inst('noun(242109-1-4)','AGM-88I')).
lf(theme('noun(242109-2-1)',obj('noun(242109-1-4)'))).
lf(quantity('noun(242109-2-1)',full)).
lf(time('noun(242109-2-1)',before('pastpart(242109-1-1)'))).
lf(inst('noun(242109-2-1)','side view')).
lf(location('noun(242109-1-6)',over('noun(242109-1-8)'))).
lf(inst('noun(242109-1-6)','A-6B')).
lf(theme('noun(242109-1-1)',obj('noun(242109-1-4)'))).
lf(inst('noun(242109-1-1)','air-to-air view')).
lf(inst('noun(242109-1-a231)',air)).
lf(inst('noun(242109-1-a230)','NWC')).
lf(theme('pastpart(242109-1-1)',obj('noun(242109-1-4)'))).
lf(source('pastpart(242109-1-1)',from('noun(242109-1-6)'))).
lf(activity('pastpart(242109-1-1)',launch)).
lf(attribute('noun(242109-1-8)',inst('noun(242109-1-a230)'))).
lf(inst('noun(242109-1-8)',range)).
::::::::::::::
```

```
242112
::::::::::::::

lf(time('noun(242112-1-4)',at('noun(242112-2-2)'))).
lf(location('noun(242112-1-4)',in('noun(242112-1-a233)'))).
lf(inst('noun(242112-1-4)','AGM-88I')).
lf(location('noun(242112-1-6)',over('noun(242112-1-8)'))).
lf(inst('noun(242112-1-6)','A-6B')).
lf(theme('noun(242112-1-1)',obj('noun(242112-1-4)'))).
lf(inst('noun(242112-1-1)','air-to-air view')).
lf(agent('prespart(242112-4-1)',obj('noun(242112-1-4)'))).
lf(source('prespart(242112-4-1)',obj('noun(242112-4-2)'))).
lf(activity('prespart(242112-4-1)',depart)).
lf(theme('noun(242112-2-2)',obj('noun(242112-1-4)'))).
lf(inst('noun(242112-2-2)',ignition)).
lf(inst('noun(242112-1-a233)',air)).
lf(inst('noun(242112-4-2)',launcher)).
lf(attribute('noun(242112-3-1)',next_to('noun(242112-1-6)'))).
lf(inst('noun(242112-3-1)',plume)).
lf(inst('noun(242112-1-a232)','NWC')).
lf(theme('pastpart(242112-1-1)',obj('noun(242112-1-4)'))).
lf(source('pastpart(242112-1-1)',from('noun(242112-1-6)'))).
lf(activity('pastpart(242112-1-1)',launch)).
lf(attribute('noun(242112-1-8)',inst('noun(242112-1-a232)'))).
lf(inst('noun(242112-1-8)',range)).
::::::::::::::
247152
::::::::::::::

lf(inst('noun(247152-2-2)','Coso Range')).
lf(quantity('noun(247152-1-4)',plural)).
lf(inst('noun(247152-1-4)','MK-82')).
lf(quantity('noun(247152-1-2)',plural(2))).
lf(location('noun(247152-1-2)',in('noun(247152-1-a234)'))).
lf(inst('noun(247152-1-2)','A-7E')).
lf(theme('noun(247152-1-1)',obj('noun(247152-1-2)'))).
lf(inst('noun(247152-1-1)','air-to-air view')).
lf(inst('noun(247152-3-2)',background)).
lf(quantity('noun(247152-3-1)',plural)).
lf(location('noun(247152-3-1)',in('noun(247152-3-2)'))).
lf(inst('noun(247152-3-1)','Sierra Nevada')).
lf(agent('prespart(247152-2-1)',obj('noun(247152-1-2)'))).
lf(location('prespart(247152-2-1)',over('noun(247152-2-2)'))).
lf(activity('prespart(247152-2-1)',fly)).
lf(theme('pastpart(247152-1-1)',obj('noun(247152-1-2)'))).
lf(object('pastpart(247152-1-1)',with('noun(247152-1-4)'))).
lf(activity('pastpart(247152-1-1)',assemble)).
lf(inst('noun(247152-1-a234)',air)).
::::::::::::::
247153
::::::::::::::

lf(inst('noun(247153-2-2)','Coso Range')).
lf(quantity('noun(247153-1-4)',plural)).
lf(inst('noun(247153-1-4)','MK-82')).
lf(quantity('noun(247153-1-2)',plural(2))).
lf(location('noun(247153-1-2)',in('noun(247153-1-a235)'))).
lf(inst('noun(247153-1-2)','A-7E')).
lf(theme('noun(247153-1-1)',obj('noun(247153-1-2)'))).
lf(inst('noun(247153-1-1)','air-to-air view')).
lf(agent('prespart(247153-2-1)',obj('noun(247153-1-2)'))).
```

```
lf(location('prespart(247153-2-1)',over('noun(247153-2-2)'))).
lf(activity('prespart(247153-2-1)',fly)).
lf(theme('pastpart(247153-1-1)',obj('noun(247153-1-2)'))).
lf(object('pastpart(247153-1-1)',with('noun(247153-1-4)'))).
lf(activity('pastpart(247153-1-1)',assemble)).
lf(inst('noun(247153-1-a235)',air)).
::::::::::::::
247155
::::::::::::::

lf(quantity('noun(247155-1-4)',plural)).
lf(inst('noun(247155-1-4)','MK-82')).
lf(quantity('noun(247155-1-2)',plural(2))).
lf(location('noun(247155-1-2)',in('noun(247155-1-a236)'))).
lf(inst('noun(247155-1-2)','A-7E')).
lf(theme('noun(247155-1-1)',obj('noun(247155-1-2)'))).
lf(inst('noun(247155-1-1)','air-to-air view')).
lf(agent('prespart(247155-2-1)',obj('noun(247155-1-2)'))).
lf(location('prespart(247155-2-1)',over('noun(247155-2-2)'))).
lf(activity('prespart(247155-2-1)',fly)).
lf(quantity('noun(247155-2-2)',plural)).
lf(inst('noun(247155-2-2)','Sierra Nevada')).
lf(theme('pastpart(247155-1-1)',obj('noun(247155-1-2)'))).
lf(object('pastpart(247155-1-1)',with('noun(247155-1-4)'))).
lf(activity('pastpart(247155-1-1)',assemble)).
lf(inst('noun(247155-1-a236)',air)).
::::::::::::::
247181
::::::::::::::

lf(inst('noun(247181-1-a237)','AIM-9R')).
lf(attribute('noun(247181-1-3)',part_of('noun(247181-1-a237)'))).
lf(inst('noun(247181-1-3)','homing device')).
lf(attribute('noun(247181-1-4)',not_part_of('noun(247181-1-3)'))).
lf(inst('noun(247181-1-4)',dome)).
::::::::::::::
247186
::::::::::::::

lf(inst('noun(247186-1-a238)','AIM-9R')).
lf(attribute('noun(247186-1-3)',part_of('noun(247186-1-a238)'))).
lf(inst('noun(247186-1-3)','homing device')).
lf(attribute('noun(247186-1-4)',part_of('noun(247186-1-3)'))).
lf(inst('noun(247186-1-4)',dome)).
::::::::::::::
247740
::::::::::::::

lf(inst('noun(247740-1-9)','G Range')).
lf(quantity('noun(247740-2-1)',full)).
lf(theme('noun(247740-2-1)',obj('noun(247740-1-3)'))).
lf(inst('noun(247740-2-1)','side view')).
lf(attribute('noun(247740-1-3)',has_part('noun(247740-1-4)'))).
lf(attribute('noun(247740-1-3)',inst('noun(247740-1-5)'))).
lf(attribute('noun(247740-1-3)',inst('noun(247740-1-2)'))).
lf(inst('noun(247740-1-3)','F/A-18A')).
lf(attribute('noun(247740-1-2)','162396')).
lf(inst('noun(247740-1-2)',bureau_no)).
lf(location('noun(247740-1-8)',over('noun(247740-1-9)'))).
lf(location('noun(247740-1-8)',on('noun(247740-1-3)'))).
lf(inst('noun(247740-1-8)','Shrike')).
```

```
lf(location('noun(247740-1-5)',on('noun(247740-1-6)'))).
lf(inst('noun(247740-1-5)','China Lake')).
lf(inst('noun(247740-1-6)',tail)).
lf(attribute('noun(247740-1-4)','105')).
lf(inst('noun(247740-1-4)',nose)).
::::::::::::::
247741
::::::::::::::

lf(inst('noun(247741-1-9)','G Range')).
lf(state('noun(247741-2-1)',close)).
lf(theme('noun(247741-2-1)',obj('noun(247741-1-3)'))).
lf(inst('noun(247741-2-1)','bottom view')).
lf(attribute('noun(247741-1-3)',has_part('noun(247741-1-4)'))).
lf(attribute('noun(247741-1-3)',inst('noun(247741-1-5)'))).
lf(attribute('noun(247741-1-3)',inst('noun(247741-1-2)'))).
lf(inst('noun(247741-1-3)','F/A-18A')).
lf(attribute('noun(247741-1-2)','162396')).
lf(inst('noun(247741-1-2)',bureau_no)).
lf(location('noun(247741-1-8)',over('noun(247741-1-9)'))).
lf(location('noun(247741-1-8)',on('noun(247741-1-3)'))).
lf(inst('noun(247741-1-8)','Shrike')).
lf(location('noun(247741-1-5)',on('noun(247741-1-6)'))).
lf(inst('noun(247741-1-5)','China Lake')).
lf(inst('noun(247741-1-6)',tail)).
lf(attribute('noun(247741-1-4)','105')).
lf(inst('noun(247741-1-4)',nose)).
::::::::::::::
248387
::::::::::::::

lf(attribute('noun(248387-1-4)','1')).
lf(theme('noun(248387-1-4)',obj('noun(248387-1-a239)'))).
lf(attribute('noun(248387-1-4)',fast)).
lf(inst('noun(248387-1-4)','cookoff test')).
lf(inst('noun(248387-1-a240)','MK-82')).
lf(attribute('noun(248387-1-a239)',part_of('noun(248387-1-a240)'))).
lf(inst('noun(248387-1-a239)','HBX')).
lf(time('noun(248387-2-1)',pretest)).
lf(inst('noun(248387-2-1)',view)).
::::::::::::::
249254
::::::::::::::

lf(inst('noun(249254-1-2)','Tomahawk')).
lf(theme('noun(249254-3-1)',obj('noun(249254-1-a241)'))).
lf(attribute('noun(249254-3-1)',round)).
lf(inst('noun(249254-3-1)',explosion)).
lf(inst('noun(249254-1-a241)','F-4')).
lf(theme('be(249254-3-1)',obj('noun(249254-3-1)'))).
lf(attribute('be(249254-3-1)',overhead)).
lf(activity('be(249254-3-1)',be)).
lf(agent('pres(249254-2-1)',obj('noun(249254-1-a241)'))).
lf(activity('pres(249254-2-1)',heat)).
lf(theme('prespart(249254-1-1)',obj('noun(249254-1-2)'))).
lf(location('prespart(249254-1-1)',over('noun(249254-1-4)'))).
lf(activity('prespart(249254-1-1)',explode)).
lf(inst('noun(249254-1-5)',revetment)).
lf(attribute('noun(249254-1-4)',isa('noun(249254-1-a241)'))).
lf(location('noun(249254-1-4)',in('noun(249254-1-5)'))).
lf(inst('noun(249254-1-4)',target)).
```

```
::::::::::::::
250629
::::::::::::::

lf(inst('noun(250629-2-a242)','Tacit/Rainbow')).
lf(attribute('noun(250629-2-4)',inst('noun(250629-2-3)'))).
lf(location('noun(250629-2-4)',in('noun(250629-2-a243)'))).
lf(inst('noun(250629-2-4)','A-3B')).
lf(attribute('noun(250629-2-3)','142630')).
lf(inst('noun(250629-2-3)',bureau_no)).
lf(theme('noun(250629-2-1)',obj('noun(250629-2-4)'))).
lf(quantity('noun(250629-2-1)',closeup)).
lf(theme('noun(250629-2-1)',obj('noun(250629-3-2)'))).
lf(inst('noun(250629-2-1)','air-to-air view')).
lf(theme('pastpart(250629-3-1)',obj('noun(250629-3-4)'))).
lf(activity('pastpart(250629-3-1)',show)).
lf(location('noun(250629-3-2)',on('noun(250629-2-4)'))).
lf(location('noun(250629-3-2)',in('noun(250629-2-a243)'))).
lf(inst('noun(250629-3-2)',front)).
lf(theme('pastpart(250629-2-1)',obj('noun(250629-2-4)'))).
lf(object('pastpart(250629-2-1)',with('noun(250629-2-6)'))).
lf(activity('pastpart(250629-2-1)',assemble)).
lf(attribute('noun(250629-2-6)',part_of('noun(250629-2-a242)'))).
lf(inst('noun(250629-2-6)','homing device')).
lf(inst('noun(250629-2-a243)',air)).
lf(attribute('noun(250629-3-4)',part_of('noun(250629-2-4)'))).
lf(inst('noun(250629-3-4)',nose)).
lf(attribute('noun(250629-1-1)','88031')).
lf(inst('noun(250629-1-1)','test plan')).
::::::::::::::
250630
::::::::::::::

lf(inst('noun(250630-2-a244)','Tacit/Rainbow')).
lf(theme('noun(250630-3-1)',obj('noun(250630-2-4)'))).
lf(inst('noun(250630-3-1)','overall view')).
lf(attribute('noun(250630-2-4)',inst('noun(250630-2-3)'))).
lf(quantity('noun(250630-2-4)',all)).
lf(location('noun(250630-2-4)',in('noun(250630-2-a245)'))).
lf(inst('noun(250630-2-4)','A-3B')).
lf(attribute('noun(250630-2-3)','142630')).
lf(inst('noun(250630-2-3)',bureau_no)).
lf(theme('noun(250630-2-1)',obj('noun(250630-2-4)'))).
lf(inst('noun(250630-2-1)','air-to-air view')).
lf(theme('pastpart(250630-2-1)',obj('noun(250630-2-4)'))).
lf(object('pastpart(250630-2-1)',with('noun(250630-2-6)'))).
lf(activity('pastpart(250630-2-1)',assemble)).
lf(attribute('noun(250630-2-6)',part_of('noun(250630-2-a244)'))).
lf(inst('noun(250630-2-6)','homing device')).
lf(inst('noun(250630-2-a245)',air)).
lf(attribute('noun(250630-1-1)','88031')).
lf(inst('noun(250630-1-1)','test plan')).
::::::::::::::
251272
::::::::::::::

lf(inst('noun(251272-4-3)','pusher sled')).
lf(attribute('noun(251272-2-a247)',owned_by('noun(251272-2-a248)'))).
lf(inst('noun(251272-2-a247)','T-45A')).
lf(inst('noun(251272-7-2)','drogue parachute')).
lf(attribute('coordinate(251272-3-1)','4597 '' N x 51 '' W')).
```

```
lf(inst('coordinate(251272-3-1)',coordinate)).
lf(inst('noun(251272-2-a248)','Naces')).
lf(attribute('noun(251272-2-6)','1')).
lf(theme('noun(251272-2-6)',obj('noun(251272-2-5)'))).
lf(inst('noun(251272-2-6)',run)).
lf(theme('noun(251272-2-5)',obj('noun(251272-2-a246)'))).
lf(inst('noun(251272-2-5)',ejection)).
lf(attribute('noun(251272-3-1)',synchronous)).
lf(location('noun(251272-3-1)',at('coordinate(251272-3-1)'))).
lf(activity('noun(251272-3-1)',launch)).
lf(theme('pastpart(251272-7-1)',obj('noun(251272-7-2)'))).
lf(activity('pastpart(251272-7-1)',expand)).
lf(location('noun(251272-6-1)',over('noun(251272-4-3)'))).
lf(inst('noun(251272-6-1)',dummy)).
lf(attribute('noun(251272-5-2)',part_of('noun(251272-2-a246)'))).
lf(quantity('noun(251272-5-2)',plural(both))).
lf(inst('noun(251272-5-2)',rocket)).
lf(inst('noun(251272-4-a249)',fire)).
lf(attribute('noun(251272-4-2)',isa('noun(251272-4-a249)'))).
lf(source('noun(251272-4-2)',from('noun(251272-4-3)'))).
lf(attribute('noun(251272-4-2)',part_of('noun(251272-5-2)'))).
lf(inst('noun(251272-4-2)',plume)).
lf(location('noun(251272-2-a246)',on('noun(251272-2-a247)'))).
lf(inst('noun(251272-2-a246)',seat)).
lf(attribute('noun(251272-1-1)','88052')).
lf(inst('noun(251272-1-1)','test plan')).
::::::::::::::
251701
::::::::::::::

lf(location('noun(251701-2-10)',on('noun(251701-2-4)'))).
lf(inst('noun(251701-2-10)','AGM-65E')).
lf(inst('noun(251701-3-3)',camera)).
lf(attribute('noun(251701-3-2)',left)).
lf(inst('noun(251701-3-2)',bank)).
lf(attribute('noun(251701-2-4)',has_part('noun(251701-2-5)'))).
lf(attribute('noun(251701-2-4)',inst('noun(251701-2-6)'))).
lf(attribute('noun(251701-2-4)',inst('noun(251701-2-3)'))).
lf(location('noun(251701-2-4)',in('noun(251701-2-a251)'))).
lf(inst('noun(251701-2-4)','F/A-18C')).
lf(quantity('noun(251701-2-12)',plural(2))).
lf(attribute('noun(251701-2-12)',inert)).
lf(location('noun(251701-2-12)',on('noun(251701-2-4)'))).
lf(inst('noun(251701-2-12)','AIM-9')).
lf(attribute('noun(251701-2-3)','163428')).
lf(inst('noun(251701-2-3)',bureau_no)).
lf(theme('noun(251701-2-1)',obj('noun(251701-2-4)'))).
lf(inst('noun(251701-2-1)','air-to-air view')).
lf(location('noun(251701-2-6)',on('noun(251701-2-7)'))).
lf(inst('noun(251701-2-6)','flying eagle')).
lf(inst('noun(251701-2-a251)',air)).
lf(theme('prespart(251701-3-1)',obj('noun(251701-3-2)'))).
lf(agent('prespart(251701-3-1)',obj('noun(251701-2-4)'))).
lf(source('prespart(251701-3-1)',from('noun(251701-3-3)'))).
lf(activity('prespart(251701-3-1)',pbuild)).
lf(attribute('noun(251701-2-a250)',part_of('noun(251701-2-10)'))).
lf(inst('noun(251701-2-a250)',laser)).
lf(inst('noun(251701-2-7)',tail)).
lf(attribute('noun(251701-2-5)','110')).
lf(inst('noun(251701-2-5)',nose)).
lf(attribute('noun(251701-1-1)','86057')).
```

```
lf(inst('noun(251701-1-1)','test plan')).
::::::::::::::
251703
::::::::::::::

lf(location('noun(251703-2-10)',on('noun(251703-2-4)'))).
lf(inst('noun(251703-2-10)','AGM-65E')).
lf(inst('noun(251703-3-3)',camera)).
lf(attribute('noun(251703-3-2)',right)).
lf(inst('noun(251703-3-2)',bank)).
lf(attribute('noun(251703-2-4)',has_part('noun(251703-2-5)'))).
lf(attribute('noun(251703-2-4)',inst('noun(251703-2-6)'))).
lf(attribute('noun(251703-2-4)',inst('noun(251703-2-3)'))).
lf(location('noun(251703-2-4)',in('noun(251703-2-a254)'))).
lf(inst('noun(251703-2-4)','F/A-18C')).
lf(quantity('noun(251703-2-12)',plural(2))).
lf(attribute('noun(251703-2-12)',inert)).
lf(location('noun(251703-2-12)',on('noun(251703-2-4)'))).
lf(inst('noun(251703-2-12)','AIM-9')).
lf(attribute('noun(251703-2-3)','163428')).
lf(inst('noun(251703-2-3)',bureau_no)).
lf(theme('noun(251703-2-1)',obj('noun(251703-2-4)'))).
lf(inst('noun(251703-2-1)','air-to-air view')).
lf(inst('noun(251703-4-3)',background)).
lf(quantity('noun(251703-4-2)',plural)).
lf(attribute('noun(251703-4-2)',inst('noun(251703-4-a253)'))).
lf(location('noun(251703-4-2)',in('noun(251703-4-3)'))).
lf(inst('noun(251703-4-2)',formation)).
lf(location('noun(251703-2-6)',on('noun(251703-2-7)'))).
lf(inst('noun(251703-2-6)','flying eagle')).
lf(inst('noun(251703-2-a254)',air)).
lf(attribute('noun(251703-4-a253)',excellent)).
lf(inst('noun(251703-4-a253)',cloud)).
lf(theme('prespart(251703-3-1)',obj('noun(251703-3-2)'))).
lf(agent('prespart(251703-3-1)',obj('noun(251703-2-4)'))).
lf(destination('prespart(251703-3-1)',to('noun(251703-3-3)'))).
lf(activity('prespart(251703-3-1)',pbuild)).
lf(attribute('noun(251703-2-a252)',part_of('noun(251703-2-10)'))).
lf(inst('noun(251703-2-a252)',laser)).
lf(inst('noun(251703-2-7)',tail)).
lf(attribute('noun(251703-2-5)','110')).
lf(inst('noun(251703-2-5)',nose)).
lf(attribute('noun(251703-1-1)','86057')).
lf(inst('noun(251703-1-1)','test plan')).
::::::::::::::
251704
::::::::::::::

lf(inst('noun(251704-3-2)','Argus Range')).
lf(location('noun(251704-2-10)',on('noun(251704-2-4)'))).
lf(inst('noun(251704-2-10)','AGM-65E')).
lf(location('noun(251704-4-1)',under('noun(251704-4-3)'))).
lf(location('noun(251704-4-1)',in('noun(251704-2-a256)'))).
lf(inst('noun(251704-4-1)','Panamint Valley')).
lf(attribute('noun(251704-2-4)',has_part('noun(251704-2-5)'))).
lf(attribute('noun(251704-2-4)',inst('noun(251704-2-6)'))).
lf(attribute('noun(251704-2-4)',inst('noun(251704-2-3)'))).
lf(location('noun(251704-2-4)',over('noun(251704-3-2)'))).
lf(location('noun(251704-2-4)',in('noun(251704-2-a256)'))).
lf(inst('noun(251704-2-4)','F/A-18C')).
lf(quantity('noun(251704-2-12)',plural(2))).
```

```
lf(attribute('noun(251704-2-12)',inert)).
lf(location('noun(251704-2-12)',on('noun(251704-2-4)'))).
lf(inst('noun(251704-2-12)','AIM-9')).
lf(attribute('noun(251704-2-3)','163428')).
lf(inst('noun(251704-2-3)',bureau_no)).
lf(theme('noun(251704-2-1)',obj('noun(251704-2-4)'))).
lf(theme('noun(251704-2-1)',obj('noun(251704-4-3)'))).
lf(theme('noun(251704-2-1)',obj('noun(251704-4-1)'))).
lf(inst('noun(251704-2-1)','air-to-air view')).
lf(location('noun(251704-2-6)',on('noun(251704-2-7)'))).
lf(inst('noun(251704-2-6)','flying eagle')).
lf(inst('noun(251704-2-a256)',air)).
lf(location('noun(251704-4-3)',in('noun(251704-2-a256)'))).
lf(inst('noun(251704-4-3)',cloud)).
lf(attribute('noun(251704-2-a255)',part_of('noun(251704-2-10)'))).
lf(inst('noun(251704-2-a255)',laser)).
lf(inst('noun(251704-2-7)',tail)).
lf(attribute('noun(251704-2-5)','110')).
lf(inst('noun(251704-2-5)',nose)).
lf(attribute('noun(251704-1-1)','86057')).
lf(inst('noun(251704-1-1)','test plan')).
::::::::::::::
251706
::::::::::::::

lf(inst('noun(251706-3-2)','Argus Range')).
lf(location('noun(251706-2-10)',on('noun(251706-2-4)'))).
lf(inst('noun(251706-2-10)','AGM-65E')).
lf(attribute('noun(251706-2-4)',has_part('noun(251706-2-5)'))).
lf(attribute('noun(251706-2-4)',inst('noun(251706-2-6)'))).
lf(attribute('noun(251706-2-4)',inst('noun(251706-2-3)'))).
lf(location('noun(251706-2-4)',over('noun(251706-3-2)'))).
lf(location('noun(251706-2-4)',in('noun(251706-2-a258)'))).
lf(inst('noun(251706-2-4)','F/A-18C')).
lf(quantity('noun(251706-2-12)',plural(2))).
lf(attribute('noun(251706-2-12)',inert)).
lf(location('noun(251706-2-12)',on('noun(251706-2-4)'))).
lf(inst('noun(251706-2-12)','AIM-9')).
lf(attribute('noun(251706-2-3)','163428')).
lf(inst('noun(251706-2-3)',bureau_no)).
lf(theme('noun(251706-2-1)',obj('noun(251706-2-4)'))).
lf(inst('noun(251706-2-1)','air-to-air view')).
lf(location('noun(251706-2-6)',on('noun(251706-2-7)'))).
lf(inst('noun(251706-2-6)','flying eagle')).
lf(inst('noun(251706-2-a258)',air)).
lf(attribute('noun(251706-4-1)',pretty)).
lf(inst('noun(251706-4-1)',cloud)).
lf(attribute('noun(251706-2-a257)',part_of('noun(251706-2-10)'))).
lf(inst('noun(251706-2-a257)',laser)).
lf(inst('noun(251706-2-7)',tail)).
lf(attribute('noun(251706-2-5)','110')).
lf(inst('noun(251706-2-5)',nose)).
lf(attribute('noun(251706-1-1)','86057')).
lf(inst('noun(251706-1-1)','test plan')).
::::::::::::::
251707
::::::::::::::

lf(location('noun(251707-2-10)',on('noun(251707-2-4)'))).
lf(inst('noun(251707-2-10)','AGM-65E')).
lf(attribute('noun(251707-2-4)',has_part('noun(251707-2-5)'))).
```

```
lf(attribute('noun(251707-2-4)',inst('noun(251707-2-6)'))).
lf(attribute('noun(251707-2-4)',inst('noun(251707-2-3)'))).
lf(location('noun(251707-2-4)',in('noun(251707-2-a14)'))).
lf(inst('noun(251707-2-4)','F/A-18C')).
lf(quantity('noun(251707-2-12)',plural(2))).
lf(attribute('noun(251707-2-12)',inert)).
lf(location('noun(251707-2-12)',on('noun(251707-2-4)'))).
lf(inst('noun(251707-2-12)','AIM-9')).
lf(attribute('noun(251707-2-3)','163428')).
lf(inst('noun(251707-2-3)',bureau_no)).
lf(theme('noun(251707-2-1)',obj('noun(251707-2-4)'))).
lf(inst('noun(251707-2-1)','air-to-air view')).
lf(attribute('noun(251707-3-2)',level)).
lf(agent('noun(251707-3-2)',obj('noun(251707-2-4)'))).
lf(activity('noun(251707-3-2)',fly)).
lf(location('noun(251707-2-6)',on('noun(251707-2-7)'))).
lf(inst('noun(251707-2-6)','flying eagle')).
lf(inst('noun(251707-2-a14)',air)).
lf(attribute('noun(251707-2-a13)',part_of('noun(251707-2-10)'))).
lf(inst('noun(251707-2-a13)',laser)).
lf(inst('noun(251707-2-7)',tail)).
lf(attribute('noun(251707-2-5)','110')).
lf(inst('noun(251707-2-5)',nose)).
lf(attribute('noun(251707-1-1)','86057')).
lf(inst('noun(251707-1-1)','test plan')).
::::::::::::::
251708
::::::::::::::

lf(inst('noun(251708-3-2)','Coso Range')).
lf(location('noun(251708-2-10)',on('noun(251708-2-4)'))).
lf(inst('noun(251708-2-10)','AGM-65E')).
lf(theme('noun(251708-4-1)',obj('noun(251708-2-4)'))).
lf(inst('noun(251708-4-1)','side view')).
lf(attribute('noun(251708-2-4)',has_part('noun(251708-2-5)'))).
lf(attribute('noun(251708-2-4)',inst('noun(251708-2-6)'))).
lf(attribute('noun(251708-2-4)',inst('noun(251708-2-3)'))).
lf(location('noun(251708-2-4)',over('noun(251708-3-2)'))).
lf(location('noun(251708-2-4)',in('noun(251708-2-a262)'))).
lf(inst('noun(251708-2-4)','F/A-18C')).
lf(quantity('noun(251708-2-12)',plural(2))).
lf(attribute('noun(251708-2-12)',inert)).
lf(location('noun(251708-2-12)',on('noun(251708-2-4)'))).
lf(inst('noun(251708-2-12)','AIM-9')).
lf(attribute('noun(251708-2-3)','163428')).
lf(inst('noun(251708-2-3)',bureau_no)).
lf(theme('noun(251708-2-1)',obj('noun(251708-2-4)'))).
lf(inst('noun(251708-2-1)','air-to-air view')).
lf(location('noun(251708-2-6)',on('noun(251708-2-7)'))).
lf(inst('noun(251708-2-6)','flying eagle')).
lf(inst('noun(251708-2-a262)',air)).
lf(attribute('noun(251708-2-a261)',part_of('noun(251708-2-10)'))).
lf(inst('noun(251708-2-a261)',laser)).
lf(inst('noun(251708-2-7)',tail)).
lf(attribute('noun(251708-2-5)','110')).
lf(inst('noun(251708-2-5)',nose)).
lf(attribute('noun(251708-1-1)','86057')).
lf(inst('noun(251708-1-1)','test plan')).
::::::::::::::
251709
::::::::::::::
```

```
lf(inst('noun(251709-3-2)','Coso Range')).
lf(location('noun(251709-2-10)',on('noun(251709-2-4)'))).
lf(inst('noun(251709-2-10)','AGM-65E')).
lf(theme('noun(251709-4-1)',obj('noun(251709-2-4)'))).
lf(attribute('noun(251709-4-1)',vertical)).
lf(inst('noun(251709-4-1)','frame view')).
lf(attribute('noun(251709-2-4)',has_part('noun(251709-2-5)'))).
lf(attribute('noun(251709-2-4)',inst('noun(251709-2-6)'))).
lf(attribute('noun(251709-2-4)',inst('noun(251709-2-3)'))).
lf(location('noun(251709-2-4)',over('noun(251709-3-2)'))).
lf(location('noun(251709-2-4)',in('noun(251709-2-a264)'))).
lf(inst('noun(251709-2-4)','F/A-18C')).
lf(quantity('noun(251709-2-12)',plural(2))).
lf(attribute('noun(251709-2-12)',inert)).
lf(location('noun(251709-2-12)',on('noun(251709-2-4)'))).
lf(inst('noun(251709-2-12)','AIM-9')).
lf(attribute('noun(251709-2-3)','163428')).
lf(inst('noun(251709-2-3)',bureau_no)).
lf(theme('noun(251709-2-1)',obj('noun(251709-2-4)'))).
lf(inst('noun(251709-2-1)','air-to-air view')).
lf(location('noun(251709-2-6)',on('noun(251709-2-7)'))).
lf(inst('noun(251709-2-6)','flying eagle')).
lf(inst('noun(251709-2-a264)',air)).
lf(attribute('noun(251709-2-a263)',part_of('noun(251709-2-10)'))).
lf(inst('noun(251709-2-a263)',laser)).
lf(inst('noun(251709-2-7)',tail)).
lf(attribute('noun(251709-2-5)','110')).
lf(inst('noun(251709-2-5)',nose)).
lf(attribute('noun(251709-1-1)','86057')).
lf(inst('noun(251709-1-1)','test plan')).
::::::::::::::
251710
::::::::::::::

lf(inst('noun(251710-3-2)','Coso Range')).
lf(location('noun(251710-2-10)',on('noun(251710-2-4)'))).
lf(inst('noun(251710-2-10)','AGM-65E')).
lf(theme('noun(251710-4-1)',obj('noun(251710-2-4)'))).
lf(quantity('noun(251710-4-1)',full)).
lf(inst('noun(251710-4-1)','frame view')).
lf(attribute('noun(251710-2-4)',has_part('noun(251710-2-5)'))).
lf(attribute('noun(251710-2-4)',inst('noun(251710-2-6)'))).
lf(attribute('noun(251710-2-4)',inst('noun(251710-2-3)'))).
lf(location('noun(251710-2-4)',over('noun(251710-3-2)'))).
lf(location('noun(251710-2-4)',in('noun(251710-2-a266)'))).
lf(inst('noun(251710-2-4)','F/A-18C')).
lf(quantity('noun(251710-2-12)',plural(2))).
lf(attribute('noun(251710-2-12)',inert)).
lf(location('noun(251710-2-12)',on('noun(251710-2-4)'))).
lf(inst('noun(251710-2-12)','AIM-9')).
lf(attribute('noun(251710-2-3)','163428')).
lf(inst('noun(251710-2-3)',bureau_no)).
lf(theme('noun(251710-2-1)',obj('noun(251710-2-4)'))).
lf(inst('noun(251710-2-1)','air-to-air view')).
lf(location('noun(251710-2-6)',on('noun(251710-2-7)'))).
lf(inst('noun(251710-2-6)','flying eagle')).
lf(inst('noun(251710-2-a266)',air)).
lf(attribute('noun(251710-2-a265)',part_of('noun(251710-2-10)'))).
lf(inst('noun(251710-2-a265)',laser)).
lf(inst('noun(251710-2-7)',tail)).
```

```
lf(attribute('noun(251710-2-5)','110')).
lf(inst('noun(251710-2-5)',nose)).
lf(attribute('noun(251710-1-1)','86057')).
lf(inst('noun(251710-1-1)','test plan')).
::::::::::::::
251855
::::::::::::::

lf(attribute('noun(251855-1-3)',has_part('noun(251855-1-4)'))).
lf(attribute('noun(251855-1-3)',inst('noun(251855-1-2)'))).
lf(inst('noun(251855-1-3)','TH-1L')).
lf(location('noun(251855-2-2)',on('noun(251855-1-3)'))).
lf(inst('noun(251855-2-2)','human being')).
lf(theme('noun(251855-2-1)',obj('noun(251855-1-3)'))).
lf(attribute('noun(251855-2-1)',left)).
lf(theme('noun(251855-2-1)',obj('noun(251855-2-2)'))).
lf(inst('noun(251855-2-1)','side view')).
lf(attribute('noun(251855-1-2)','157834')).
lf(inst('noun(251855-1-2)',bureau_no)).
lf(agent('prespart(251855-1-1)',obj('noun(251855-1-3)'))).
lf(activity('prespart(251855-1-1)',depart)).
lf(attribute('noun(251855-1-4)','14')).
lf(inst('noun(251855-1-4)',nose)).
::::::::::::::
252492
::::::::::::::

lf(quantity('noun(252492-2-a267)',plural(2))).
lf(inst('noun(252492-2-a267)','AIM-9M')).
lf(theme('noun(252492-4-1)',obj('noun(252492-2-a267)'))).
lf(quantity('noun(252492-4-1)',full)).
lf(location('noun(252492-4-1)',under('noun(252492-2-5)'))).
lf(inst('noun(252492-4-1)','side view')).
lf(inst('noun(252492-2-5)','F/A-18A')).
lf(inst('noun(252492-2-7)','BQM-34S')).
lf(theme('prespart(252492-3-1)',obj('noun(252492-2-a267)'))).
lf(location('prespart(252492-3-1)',on('noun(252492-2-5)'))).
lf(activity('prespart(252492-3-1)',assemble)).
lf(theme('noun(252492-2-3)',obj('noun(252492-2-a267)'))).
lf(destination('noun(252492-2-3)',at('noun(252492-2-7)'))).
lf(source('noun(252492-2-3)',from('noun(252492-2-5)'))).
lf(activity('noun(252492-2-3)',launch)).
lf(attribute('noun(252492-1-1)','87209')).
lf(inst('noun(252492-1-1)','test plan')).
::::::::::::::
252494
::::::::::::::

lf(inst('noun(252494-2-a268)','AIM-9M')).
lf(inst('noun(252494-2-5)','F/A-18A')).
lf(inst('noun(252494-2-7)','BQM-34S')).
lf(attribute('noun(252494-4-2)',large)).
lf(attribute('noun(252494-4-2)',part_of('noun(252494-2-a268)'))).
lf(inst('noun(252494-4-2)',plume)).
lf(theme('pastpart(252494-3-1)',obj('noun(252494-2-a268)'))).
lf(location('pastpart(252494-3-1)',before('noun(252494-2-5)'))).
lf(source('pastpart(252494-3-1)',from('noun(252494-2-5)'))).
lf(destination('pastpart(252494-3-1)',at('noun(252494-2-7)'))).
lf(activity('pastpart(252494-3-1)',launch)).
lf(attribute('noun(252494-1-1)','87209')).
lf(inst('noun(252494-1-1)','test plan')).
```

```
::::::::::::::
252496
::::::::::::::

lf(inst('noun(252496-5-a270)',smoke)).
lf(inst('noun(252496-2-a269)','AIM-9M')).
lf(inst('noun(252496-2-5)','F/A-18A')).
lf(inst('noun(252496-2-7)','BQM-34S')).
lf(agent('prespart(252496-5-1)',obj('noun(252496-5-2)'))).
lf(location('prespart(252496-5-1)',behind('noun(252496-2-5)'))).
lf(activity('prespart(252496-5-1)',expand)).
lf(attribute('noun(252496-5-2)',inst('noun(252496-5-a270)'))).
lf(inst('noun(252496-5-2)',trail)).
lf(theme('pastpart(252496-3-1)',obj('noun(252496-2-a269)'))).
lf(location('pastpart(252496-3-1)',before('noun(252496-2-5)'))).
lf(source('pastpart(252496-3-1)',from('noun(252496-2-5)'))).
lf(destination('pastpart(252496-3-1)',at('noun(252496-2-7)'))).
lf(activity('pastpart(252496-3-1)',launch)).
lf(attribute('noun(252496-1-1)','87209')).
lf(inst('noun(252496-1-1)','test plan')).
::::::::::::::
253959
::::::::::::::

lf(attribute('noun(253959-4-1)',right)).
lf(quantity('noun(253959-4-1)',closeup)).
lf(theme('noun(253959-4-1)',obj('noun(253959-2-4)'))).
lf(inst('noun(253959-4-1)','side view')).
lf(inst('noun(253959-3-5)','BTV')).
lf(attribute('noun(253959-2-4)',has_part('noun(253959-2-5)'))).
lf(attribute('noun(253959-2-4)',inst('noun(253959-2-3)'))).
lf(location('noun(253959-2-4)',in('noun(253959-2-a272)'))).
lf(inst('noun(253959-2-4)','A-7C')).
lf(attribute('noun(253959-2-3)','156738')).
lf(inst('noun(253959-2-3)',bureau_no)).
lf(theme('noun(253959-2-1)',obj('noun(253959-2-4)'))).
lf(inst('noun(253959-2-1)','air-to-air view')).
lf(attribute('noun(253959-3-a271)',part_of('noun(253959-1-2)'))).
lf(inst('noun(253959-3-a271)','fiber optic')).
lf(inst('noun(253959-1-2)','Skyray')).
lf(inst('noun(253959-2-a272)',air)).
lf(theme('be(253959-3-1)',obj('noun(253959-1-2)'))).
lf(theme('be(253959-3-1)',obj('noun(253959-3-5)'))).
lf(location('be(253959-3-1)',under('noun(253959-3-7)'))).
lf(activity('be(253959-3-1)',be)).
lf(inst('noun(253959-3-7)',wing)).
lf(attribute('noun(253959-2-5)','700')).
lf(inst('noun(253959-2-5)',nose)).
::::::::::::::
253960
::::::::::::::

lf(location('noun(253960-4-1)',at('noun(253960-4-2)'))).
lf(inst('noun(253960-4-1)','HS camera')).
lf(location('noun(253960-5-2)',at('noun(253960-5-4)'))).
lf(inst('noun(253960-5-2)','BTV')).
lf(attribute('noun(253960-2-4)',has_part('noun(253960-2-5)'))).
lf(attribute('noun(253960-2-4)',inst('noun(253960-2-3)'))).
lf(location('noun(253960-2-4)',in('noun(253960-2-a273)'))).
lf(inst('noun(253960-2-4)','A-7C')).
lf(attribute('noun(253960-2-3)','156738')).
```

```
lf(inst('noun(253960-2-3)',bureau_no)).
lf(theme('noun(253960-2-1)',obj('noun(253960-2-4)'))).
lf(location('noun(253960-2-1)',under('noun(253960-2-4)'))).
lf(inst('noun(253960-2-1)','air-to-air view')).
lf(location('noun(253960-1-2)',at('noun(253960-6-3)'))).
lf(inst('noun(253960-1-2)','Skyray')).
lf(inst('noun(253960-2-a273)',air)).
lf(attribute('noun(253960-6-3)','8')).
lf(inst('noun(253960-6-3)',station)).
lf(attribute('noun(253960-5-4)','7')).
lf(inst('noun(253960-5-4)',station)).
lf(attribute('noun(253960-4-2)','6')).
lf(inst('noun(253960-4-2)',station)).
lf(attribute('noun(253960-2-5)','700')).
lf(inst('noun(253960-2-5)',nose)).
::::::::::::::
253961
::::::::::::::

lf(location('noun(253961-4-1)',at('noun(253961-4-2)'))).
lf(inst('noun(253961-4-1)','HS camera')).
lf(location('noun(253961-5-2)',at('noun(253961-5-4)'))).
lf(inst('noun(253961-5-2)','BTV')).
lf(quantity('noun(253961-3-1)',full)).
lf(theme('noun(253961-3-1)',obj('noun(253961-2-4)'))).
lf(inst('noun(253961-3-1)','side view')).
lf(attribute('noun(253961-2-4)',has_part('noun(253961-2-5)'))).
lf(attribute('noun(253961-2-4)',inst('noun(253961-2-3)'))).
lf(location('noun(253961-2-4)',in('noun(253961-2-a274)'))).
lf(inst('noun(253961-2-4)','A-7C')).
lf(attribute('noun(253961-2-3)','156738')).
lf(inst('noun(253961-2-3)',bureau_no)).
lf(theme('noun(253961-2-1)',obj('noun(253961-2-4)'))).
lf(inst('noun(253961-2-1)','air-to-air view')).
lf(inst('noun(253961-7-3)',background)).
lf(inst('noun(253961-7-2)',valley)).
lf(location('noun(253961-7-1)',in('noun(253961-7-3)'))).
lf(location('noun(253961-7-1)',over('noun(253961-7-2)'))).
lf(inst('noun(253961-7-1)',haze)).
lf(location('noun(253961-1-2)',at('noun(253961-6-3)'))).
lf(inst('noun(253961-1-2)','Skyray')).
lf(inst('noun(253961-2-a274)',air)).
lf(attribute('noun(253961-6-3)','8')).
lf(inst('noun(253961-6-3)',station)).
lf(attribute('noun(253961-5-4)','7')).
lf(inst('noun(253961-5-4)',station)).
lf(attribute('noun(253961-4-2)','6')).
lf(inst('noun(253961-4-2)',station)).
lf(attribute('noun(253961-2-5)','700')).
lf(inst('noun(253961-2-5)',nose)).
::::::::::::::
253962
::::::::::::::

lf(location('noun(253962-4-1)',at('noun(253962-4-2)'))).
lf(inst('noun(253962-4-1)','HS camera')).
lf(attribute('noun(253962-7-2)',inert)).
lf(location('noun(253962-7-2)',on('noun(253962-7-3)'))).
lf(inst('noun(253962-7-2)','MK-82')).
lf(location('noun(253962-5-2)',at('noun(253962-5-4)'))).
lf(inst('noun(253962-5-2)','BTV')).
```

```
lf(quantity('noun(253962-3-1)',full)).
lf(theme('noun(253962-3-1)',obj('noun(253962-2-4)'))).
lf(inst('noun(253962-3-1)','side view')).
lf(attribute('noun(253962-2-4)',has_part('noun(253962-2-5)'))).
lf(attribute('noun(253962-2-4)',inst('noun(253962-2-3)'))).
lf(location('noun(253962-2-4)',in('noun(253962-2-a275)'))).
lf(inst('noun(253962-2-4)','A-7C')).
lf(attribute('noun(253962-2-3)','156738')).
lf(inst('noun(253962-2-3)',bureau_no)).
lf(theme('noun(253962-2-1)',obj('noun(253962-2-4)'))).
lf(inst('noun(253962-2-1)','air-to-air view')).
lf(location('noun(253962-1-2)',at('noun(253962-6-3)'))).
lf(inst('noun(253962-1-2)','Skyray')).
lf(inst('noun(253962-2-a275)',air)).
lf(attribute('noun(253962-7-3)',left)).
lf(inst('noun(253962-7-3)',wing)).
lf(attribute('noun(253962-6-3)','8')).
lf(inst('noun(253962-6-3)',station)).
lf(attribute('noun(253962-5-4)','7')).
lf(inst('noun(253962-5-4)',station)).
lf(attribute('noun(253962-4-2)','6')).
lf(inst('noun(253962-4-2)',station)).
lf(attribute('noun(253962-2-5)','700')).
lf(inst('noun(253962-2-5)',nose)).
::::::::::::::
255577
::::::::::::::

lf(location('noun(255577-2-a276)',on('noun(255577-3-7)'))).
lf(location('noun(255577-2-a276)',on('noun(255577-3-3)'))).
lf(inst('noun(255577-2-a276)','AIM-9M')).
lf(attribute('noun(255577-3-3)',has_part('noun(255577-3-4)'))).
lf(attribute('noun(255577-3-3)',inst('noun(255577-3-2)'))).
lf(inst('noun(255577-3-3)','F/A-18A')).
lf(attribute('noun(255577-3-2)','161713')).
lf(inst('noun(255577-3-2)',bureau_no)).
lf(theme('noun(255577-2-3)',obj('noun(255577-2-a276)'))).
lf(location('noun(255577-2-3)',in('noun(255577-2-a277)'))).
lf(inst('noun(255577-2-3)','separation test')).
lf(theme('noun(255577-2-1)',obj('noun(255577-2-3)'))).
lf(inst('noun(255577-2-1)','air-to-air view')).
lf(theme('pastpart(255577-4-1)',obj('noun(255577-4-1)'))).
lf(source('pastpart(255577-4-1)',from('noun(255577-4-2)'))).
lf(activity('pastpart(255577-4-1)',disassemble)).
lf(quantity('noun(255577-4-2)',plural)).
lf(inst('noun(255577-4-2)','tail fin')).
lf(inst('noun(255577-2-a277)',air)).
lf(inst('noun(255577-4-1)',fuel)).
lf(attribute('noun(255577-3-7)',right)).
lf(inst('noun(255577-3-7)',wingtip)).
lf(attribute('noun(255577-3-4)','101')).
lf(inst('noun(255577-3-4)',nose)).
lf(attribute('noun(255577-1-1)','89053')).
lf(inst('noun(255577-1-1)','test plan')).
::::::::::::::
255578
::::::::::::::

lf(location('noun(255578-2-a278)',on('noun(255578-3-7)'))).
lf(location('noun(255578-2-a278)',on('noun(255578-3-3)'))).
lf(inst('noun(255578-2-a278)','AIM-9M')).
```

```
lf(attribute('noun(255578-3-3)',has_part('noun(255578-3-4)'))).
lf(attribute('noun(255578-3-3)',inst('noun(255578-3-2)'))).
lf(inst('noun(255578-3-3)','F/A-18A')).
lf(attribute('noun(255578-3-2)','161713')).
lf(inst('noun(255578-3-2)',bureau_no)).
lf(theme('noun(255578-2-3)',obj('noun(255578-2-a278)'))).
lf(location('noun(255578-2-3)',in('noun(255578-2-a279)'))).
lf(inst('noun(255578-2-3)','separation test')).
lf(theme('noun(255578-2-1)',obj('noun(255578-2-3)'))).
lf(inst('noun(255578-2-1)','air-to-air view')).
lf(inst('noun(255578-5-2)',background)).
lf(location('noun(255578-5-1)',in('noun(255578-5-2)'))).
lf(inst('noun(255578-5-1)','Searles Lake')).
lf(agent('prespart(255578-4-1)',obj('noun(255578-4-1)'))).
lf(source('prespart(255578-4-1)',from('noun(255578-4-2)'))).
lf(activity('prespart(255578-4-1)',disassemble)).
lf(quantity('noun(255578-4-2)',plural)).
lf(inst('noun(255578-4-2)','tail fin')).
lf(inst('noun(255578-2-a279)',air)).
lf(inst('noun(255578-4-1)',fuel)).
lf(attribute('noun(255578-3-7)',right)).
lf(inst('noun(255578-3-7)',wingtip)).
lf(attribute('noun(255578-3-4)','101')).
lf(inst('noun(255578-3-4)',nose)).
lf(attribute('noun(255578-1-1)','89053')).
lf(inst('noun(255578-1-1)','test plan')).
::::::::::::::
255580
::::::::::::::

lf(location('noun(255580-2-a280)',on('noun(255580-3-7)'))).
lf(location('noun(255580-2-a280)',on('noun(255580-3-3)'))).
lf(inst('noun(255580-2-a280)','AIM-9M')).
lf(attribute('noun(255580-3-3)',has_part('noun(255580-3-4)'))).
lf(attribute('noun(255580-3-3)',inst('noun(255580-3-2)'))).
lf(inst('noun(255580-3-3)','F/A-18A')).
lf(attribute('noun(255580-3-2)','161713')).
lf(inst('noun(255580-3-2)',bureau_no)).
lf(theme('noun(255580-2-3)',obj('noun(255580-2-a280)'))).
lf(location('noun(255580-2-3)',in('noun(255580-2-a281)'))).
lf(inst('noun(255580-2-3)','separation test')).
lf(theme('noun(255580-2-1)',obj('noun(255580-2-3)'))).
lf(inst('noun(255580-2-1)','air-to-air view')).
lf(inst('noun(255580-4-2)',background)).
lf(location('noun(255580-4-1)',in('noun(255580-4-2)'))).
lf(inst('noun(255580-4-1)','Ridgecrest')).
lf(inst('noun(255580-2-a281)',air)).
lf(attribute('noun(255580-3-7)',right)).
lf(inst('noun(255580-3-7)',wingtip)).
lf(attribute('noun(255580-3-4)','101')).
lf(inst('noun(255580-3-4)',nose)).
lf(attribute('noun(255580-1-1)','89053')).
lf(inst('noun(255580-1-1)','test plan')).
::::::::::::::
255655
::::::::::::::

lf(attribute('noun(255655-1-4)',inst('noun(255655-1-5)'))).
lf(inst('noun(255655-1-4)','FSQ-12')).
lf(attribute('noun(255655-1-5)','031')).
lf(inst('noun(255655-1-5)',serial_no)).
```

```
lf(attribute('noun(255655-2-3)',inst('noun(255655-2-a282)'))).
lf(theme('noun(255655-2-3)',obj('noun(255655-1-4)'))).
lf(destination('noun(255655-2-3)',at('noun(255655-2-4)'))).
lf(inst('noun(255655-2-3)',disassembly)).
lf(time('noun(255655-2-a282)',post)).
lf(activity('noun(255655-2-a282)',launch)).
lf(attribute('noun(255655-2-4)','69')).
lf(inst('noun(255655-2-4)',building)).
::::::::::::::
256393
::::::::::::::

lf(inst('noun(256393-1-4)','AIM-9L')).
lf(inst('noun(256393-1-2)','AIM-9R')).
lf(quantity('noun(256393-4-1)',closeup)).
lf(theme('noun(256393-4-1)',obj('noun(256393-1-7)'))).
lf(inst('noun(256393-4-1)','front view')).
lf(attribute('noun(256393-1-7)',has_part('noun(256393-1-8)'))).
lf(attribute('noun(256393-1-7)',has_part('noun(256393-1-9)'))).
lf(attribute('noun(256393-1-7)',inst('noun(256393-1-6)'))).
lf(inst('noun(256393-1-7)','F/A-18A')).
lf(attribute('noun(256393-1-6)','162396')).
lf(inst('noun(256393-1-6)',bureau_no)).
lf(theme('pastpart(256393-1-1)',obj('noun(256393-1-2)'))).
lf(theme('pastpart(256393-1-1)',obj('noun(256393-1-4)'))).
lf(location('pastpart(256393-1-1)',on('noun(256393-1-7)'))).
lf(location('pastpart(256393-1-1)',on('noun(256393-2-2)'))).
lf(activity('pastpart(256393-1-1)',assemble)).
lf(inst('noun(256393-1-10)','flying eagle')).
lf(attribute('noun(256393-2-2)',outboard)).
lf(attribute('noun(256393-2-2)',inboard)).
lf(inst('noun(256393-2-2)',pylon)).
lf(attribute('noun(256393-1-9)','5')).
lf(attribute('noun(256393-1-9)',inst('noun(256393-1-10)'))).
lf(inst('noun(256393-1-9)',tail)).
lf(attribute('noun(256393-1-8)','105')).
lf(inst('noun(256393-1-8)',nose)).
::::::::::::::
256394
::::::::::::::

lf(inst('noun(256394-1-4)','AIM-9L')).
lf(quantity('noun(256394-1-2)',plural)).
lf(inst('noun(256394-1-2)','AIM-9R')).
lf(quantity('noun(256394-4-1)','1/4')).
lf(quantity('noun(256394-4-1)',closeup)).
lf(theme('noun(256394-4-1)',obj('noun(256394-4-2)'))).
lf(inst('noun(256394-4-1)','front view')).
lf(attribute('noun(256394-1-7)',has_part('noun(256394-1-8)'))).
lf(attribute('noun(256394-1-7)',has_part('noun(256394-1-9)'))).
lf(attribute('noun(256394-1-7)',inst('noun(256394-1-6)'))).
lf(inst('noun(256394-1-7)','F/A-18A')).
lf(attribute('noun(256394-1-6)','162396')).
lf(inst('noun(256394-1-6)',bureau_no)).
lf(location('noun(256394-4-2)',on('noun(256394-1-2)'))).
lf(inst('noun(256394-4-2)',front)).
lf(theme('pastpart(256394-1-1)',obj('noun(256394-1-2)'))).
lf(theme('pastpart(256394-1-1)',obj('noun(256394-1-4)'))).
lf(location('pastpart(256394-1-1)',on('noun(256394-1-7)'))).
lf(location('pastpart(256394-1-1)',on('noun(256394-2-2)'))).
lf(activity('pastpart(256394-1-1)',assemble)).
```

```
lf(inst('noun(256394-1-10)','flying eagle')).
lf(attribute('noun(256394-2-2)',outboard)).
lf(attribute('noun(256394-2-2)',inboard)).
lf(inst('noun(256394-2-2)',pylon)).
lf(attribute('noun(256394-1-9)','5')).
lf(attribute('noun(256394-1-9)',inst('noun(256394-1-10)'))).
lf(inst('noun(256394-1-9)',tail)).
lf(attribute('noun(256394-1-8)','105')).
lf(inst('noun(256394-1-8)',nose)).
::::::::::::::
256395
::::::::::::::

lf(inst('noun(256395-1-4)','AIM-9L')).
lf(quantity('noun(256395-1-2)',plural)).
lf(inst('noun(256395-1-2)','AIM-9R')).
lf(attribute('noun(256395-1-7)',has_part('noun(256395-1-8)'))).
lf(attribute('noun(256395-1-7)',has_part('noun(256395-1-9)'))).
lf(attribute('noun(256395-1-7)',inst('noun(256395-1-6)'))).
lf(inst('noun(256395-1-7)','F/A-18A')).
lf(attribute('noun(256395-1-6)','162396')).
lf(inst('noun(256395-1-6)',bureau_no)).
lf(location('noun(256395-4-3)',on('noun(256395-1-7)'))).
lf(inst('noun(256395-4-3)',rear)).
lf(quantity('noun(256395-4-1)',full)).
lf(theme('noun(256395-4-1)',obj('noun(256395-1-2)'))).
lf(theme('noun(256395-4-1)',obj('noun(256395-4-3)'))).
lf(inst('noun(256395-4-1)',view)).
lf(theme('pastpart(256395-1-1)',obj('noun(256395-1-2)'))).
lf(theme('pastpart(256395-1-1)',obj('noun(256395-1-4)'))).
lf(location('pastpart(256395-1-1)',on('noun(256395-1-7)'))).
lf(location('pastpart(256395-1-1)',on('noun(256395-2-2)'))).
lf(activity('pastpart(256395-1-1)',assemble)).
lf(inst('noun(256395-1-10)','flying eagle')).
lf(attribute('noun(256395-2-2)',outboard)).
lf(attribute('noun(256395-2-2)',inboard)).
lf(inst('noun(256395-2-2)',pylon)).
lf(attribute('noun(256395-1-9)','5')).
lf(attribute('noun(256395-1-9)',inst('noun(256395-1-10)'))).
lf(inst('noun(256395-1-9)',tail)).
lf(attribute('noun(256395-1-8)','105')).
lf(inst('noun(256395-1-8)',nose)).
::::::::::::::
256979
::::::::::::::

lf(attribute('noun(256979-3-1)',right)).
lf(quantity('noun(256979-3-1)','3/4')).
lf(theme('noun(256979-3-1)',obj('noun(256979-1-2)'))).
lf(inst('noun(256979-3-1)','front view')).
lf(attribute('noun(256979-2-3)',has_part('noun(256979-2-4)'))).
lf(attribute('noun(256979-2-3)',inst('noun(256979-2-5)'))).
lf(attribute('noun(256979-2-3)',inst('noun(256979-2-2)'))).
lf(inst('noun(256979-2-3)','F/A-18A')).
lf(attribute('noun(256979-2-2)','162396')).
lf(inst('noun(256979-2-2)',bureau_no)).
lf(location('noun(256979-2-5)',on('noun(256979-2-6)'))).
lf(inst('noun(256979-2-5)','China Lake')).
lf(attribute('noun(256979-1-2)','RTV')).
lf(location('noun(256979-1-2)',on('noun(256979-2-3)'))).
lf(inst('noun(256979-1-2)','Skyray')).
```

```
lf(inst('noun(256979-2-6)',tail)).
lf(attribute('noun(256979-2-4)','105')).
lf(inst('noun(256979-2-4)',nose)).
::::::::::::::
256999
::::::::::::::

lf(quantity('noun(256999-3-1)','3/4')).
lf(quantity('noun(256999-3-1)',closeup)).
lf(theme('noun(256999-3-1)',obj('noun(256999-1-2)'))).
lf(inst('noun(256999-3-1)','rear view')).
lf(attribute('noun(256999-2-3)',has_part('noun(256999-2-4)'))).
lf(attribute('noun(256999-2-3)',inst('noun(256999-2-5)'))).
lf(attribute('noun(256999-2-3)',inst('noun(256999-2-2)'))).
lf(inst('noun(256999-2-3)','F/A-18A')).
lf(attribute('noun(256999-2-2)','162396')).
lf(inst('noun(256999-2-2)',bureau_no)).
lf(inst('noun(256999-5-a284)',gun)).
lf(quantity('noun(256999-5-4)',plural)).
lf(attribute('noun(256999-5-4)',part_of('noun(256999-5-a284)'))).
lf(inst('noun(256999-5-4)',end)).
lf(quantity('noun(256999-4-2)',plural)).
lf(attribute('noun(256999-4-2)',inst('noun(256999-4-a283)'))).
lf(location('noun(256999-4-2)',behind('noun(256999-2-3)'))).
lf(inst('noun(256999-4-2)',formation)).
lf(location('noun(256999-2-5)',on('noun(256999-2-6)'))).
lf(inst('noun(256999-2-5)','China Lake')).
lf(attribute('noun(256999-1-2)','RTV')).
lf(location('noun(256999-1-2)',at('noun(256999-5-4)'))).
lf(location('noun(256999-1-2)',on('noun(256999-2-3)'))).
lf(inst('noun(256999-1-2)','Skyray')).
lf(attribute('noun(256999-4-a283)',beautiful)).
lf(inst('noun(256999-4-a283)',cloud)).
lf(inst('noun(256999-2-6)',tail)).
lf(attribute('noun(256999-2-4)','105')).
lf(inst('noun(256999-2-4)',nose)).
::::::::::::::
257009
::::::::::::::

lf(quantity('noun(257009-3-1)',full)).
lf(theme('noun(257009-3-1)',obj('noun(257009-1-4)'))).
lf(inst('noun(257009-3-1)','side view')).
lf(attribute('noun(257009-2-2)',special)).
lf(inst('noun(257009-2-2)',luminaire)).
lf(attribute('noun(257009-1-4)','night attack')).
lf(attribute('noun(257009-1-4)',owned_by('noun(257009-1-9)'))).
lf(attribute('noun(257009-1-4)',has_part('noun(257009-1-5)'))).
lf(attribute('noun(257009-1-4)',has_part('noun(257009-1-6)'))).
lf(attribute('noun(257009-1-4)',inst('noun(257009-1-3)'))).
lf(inst('noun(257009-1-4)','AV-8B')).
lf(attribute('noun(257009-1-3)','162966')).
lf(inst('noun(257009-1-3)',bureau_no)).
lf(inst('noun(257009-2-1)',sunset)).
lf(inst('noun(257009-1-8)','night hawk')).
lf(inst('noun(257009-1-9)','MAD')).
lf(attribute('noun(257009-1-6)','87')).
lf(attribute('noun(257009-1-6)',inst('noun(257009-1-8)'))).
lf(inst('noun(257009-1-6)',nose)).
lf(attribute('noun(257009-1-5)','162966')).
lf(inst('noun(257009-1-5)',tail)).
```

```
::::::::::::::
257019
::::::::::::::

lf(quantity('noun(257019-3-1)',closeup)).
lf(theme('noun(257019-3-1)',obj('noun(257019-1-6)'))).
lf(theme('noun(257019-3-1)',obj('noun(257019-3-3)'))).
lf(inst('noun(257019-3-1)','side view')).
lf(attribute('noun(257019-2-2)',special)).
lf(inst('noun(257019-2-2)',luminaire)).
lf(attribute('noun(257019-1-4)','night attack')).
lf(attribute('noun(257019-1-4)',owned_by('noun(257019-1-9)'))).
lf(attribute('noun(257019-1-4)',has_part('noun(257019-1-5)'))).
lf(attribute('noun(257019-1-4)',has_part('noun(257019-1-6)'))).
lf(attribute('noun(257019-1-4)',inst('noun(257019-1-3)'))).
lf(inst('noun(257019-1-4)','AV-8B')).
lf(attribute('noun(257019-1-3)','162966')).
lf(inst('noun(257019-1-3)',bureau_no)).
lf(inst('noun(257019-2-1)',sunset)).
lf(inst('noun(257019-1-8)','night hawk')).
lf(inst('noun(257019-1-9)','MAD')).
lf(inst('noun(257019-3-3)',cockpit)).
lf(attribute('noun(257019-1-6)','87')).
lf(attribute('noun(257019-1-6)',inst('noun(257019-1-8)'))).
lf(inst('noun(257019-1-6)',nose)).
lf(attribute('noun(257019-1-5)','162966')).
lf(inst('noun(257019-1-5)',tail)).
::::::::::::::
257055
::::::::::::::

lf(attribute('noun(257055-2-3)',inst('noun(257055-2-a285)'))).
lf(inst('noun(257055-2-3)','Benjes')).
lf(inst('noun(257055-1-5)','AIM-9R')).
lf(inst('noun(257055-2-a285)','Commander')).
lf(inst('noun(257055-1-3)','F/A-18A')).
lf(time('noun(257055-3-3)',on('date(257055-3-1)'))).
lf(inst('noun(257055-3-3)','QF-86')).
lf(attribute('date(257055-3-1)','12-apr-1989')).
lf(inst('date(257055-3-1)',date)).
lf(attribute('noun(257055-1-1)',graphics)).
lf(theme('noun(257055-1-1)',obj('noun(257055-1-3)'))).
lf(inst('noun(257055-1-1)',composite)).
lf(theme('noun(257055-3-1)',obj('noun(257055-1-3)'))).
lf(theme('noun(257055-3-1)',obj('noun(257055-3-3)'))).
lf(activity('noun(257055-3-1)',hit)).
lf(theme('pastpart(257055-2-1)',obj('noun(257055-1-3)'))).
lf(agent('pastpart(257055-2-1)',obj('noun(257055-2-3)'))).
lf(activity('pastpart(257055-2-1)',influence)).
lf(theme('prespart(257055-1-1)',obj('noun(257055-1-5)'))).
lf(source('prespart(257055-1-1)',from('noun(257055-1-3)'))).
lf(activity('prespart(257055-1-1)',launch)).
::::::::::::::
257110
::::::::::::::

lf(inst('noun(257110-3-a288)',pilot)).
lf(attribute('noun(257110-3-6)',inst('noun(257110-3-a287)'))).
lf(attribute('noun(257110-3-6)','back seat')).
lf(inst('noun(257110-3-6)','Weston')).
lf(attribute('noun(257110-3-3)',inst('noun(257110-3-a287)'))).
```

```
lf(attribute('noun(257110-3-3)',inst('noun(257110-3-a288)'))).
lf(inst('noun(257110-3-3)','Rzeszotko')).
lf(quantity('noun(257110-4-1)','3/4')).
lf(attribute('noun(257110-4-1)',excellent)).
lf(theme('noun(257110-4-1)',obj('noun(257110-1-5)'))).
lf(theme('noun(257110-4-1)',obj('noun(257110-4-4)'))).
lf(inst('noun(257110-4-1)','front view')).
lf(inst('noun(257110-3-a287)','Lieutenant')).
lf(inst('noun(257110-1-a286)','VX-5')).
lf(attribute('noun(257110-1-5)','night attack')).
lf(attribute('noun(257110-1-5)',owned_by('noun(257110-1-a286)'))).
lf(attribute('noun(257110-1-5)',has_part('noun(257110-1-6)'))).
lf(attribute('noun(257110-1-5)',inst('noun(257110-1-7)'))).
lf(attribute('noun(257110-1-5)',inst('noun(257110-1-3)'))).
lf(inst('noun(257110-1-5)','A-6E')).
lf(location('noun(257110-1-7)',on('noun(257110-1-8)'))).
lf(inst('noun(257110-1-7)','XE')).
lf(attribute('noun(257110-1-3)','161084')).
lf(inst('noun(257110-1-3)',bureau_no)).
lf(quantity('noun(257110-4-4)',plural)).
lf(attribute('noun(257110-4-4)',inst('noun(257110-4-a289)'))).
lf(inst('noun(257110-4-4)',formation)).
lf(quantity('noun(257110-2-2)',plural)).
lf(inst('noun(257110-2-2)','Sierra Nevada')).
lf(location('noun(257110-2-1)',over('noun(257110-2-2)'))).
lf(inst('noun(257110-2-1)',sunset)).
lf(theme('noun(257110-1-9)',obj('noun(257110-1-5)'))).
lf(inst('noun(257110-1-9)',takeoff)).
lf(inst('noun(257110-4-a289)',cloud)).
lf(agent('prespart(257110-1-1)',obj('noun(257110-1-5)'))).
lf(goal('prespart(257110-1-1)',for('noun(257110-1-9)'))).
lf(activity('prespart(257110-1-1)',prepare)).
lf(inst('noun(257110-1-8)',tail)).
lf(attribute('noun(257110-1-6)','22')).
lf(inst('noun(257110-1-6)',nose)).
::::::::::::::
257135
::::::::::::::

lf(location('noun(257135-2-a293)',in('noun(257135-3-3)'))).
lf(inst('noun(257135-2-a293)',pilot)).
lf(attribute('noun(257135-2-3)',inst('noun(257135-2-a292)'))).
lf(attribute('noun(257135-2-3)',inst('noun(257135-2-a293)'))).
lf(inst('noun(257135-2-3)','Antonio')).
lf(attribute('noun(257135-3-1)',left)).
lf(state('noun(257135-3-1)',close)).
lf(theme('noun(257135-3-1)',obj('noun(257135-2-a293)'))).
lf(inst('noun(257135-3-1)','side view')).
lf(inst('noun(257135-2-a292)','Commander')).
lf(inst('noun(257135-1-a290)','VX-5')).
lf(attribute('noun(257135-1-4)','night attack')).
lf(attribute('noun(257135-1-4)',owned_by('noun(257135-1-a290)'))).
lf(attribute('noun(257135-1-4)',has_part('noun(257135-1-5)'))).
lf(attribute('noun(257135-1-4)',has_part('noun(257135-1-6)'))).
lf(inst('noun(257135-1-4)','F/A-18A')).
lf(inst('noun(257135-2-a291)','night vision')).
lf(attribute('noun(257135-2-5)',inst('noun(257135-2-a291)'))).
lf(inst('noun(257135-2-5)',goggles)).
lf(inst('noun(257135-3-3)',cockpit)).
lf(agent('pverb(257135-2-a294)',obj('noun(257135-2-3)'))).
lf(theme('pverb(257135-2-a294)',obj('noun(257135-2-5)'))).
```

```
lf(activity('pverb(257135-2-a294)',wear)).
lf(attribute('noun(257135-1-6)','34')).
lf(inst('noun(257135-1-6)',tail)).
lf(attribute('noun(257135-1-5)','34')).
lf(inst('noun(257135-1-5)',nose)).
::::::::::::::
257274
::::::::::::::

lf(theme('noun(257274-4-1)',obj('noun(257274-2-2)'))).
lf(attribute('noun(257274-4-1)',right)).
lf(attribute('noun(257274-4-1)',overhead)).
lf(inst('noun(257274-4-1)','side view')).
lf(inst('noun(257274-3-3)','AGM-88')).
lf(inst('noun(257274-2-a297)','VX-5')).
lf(attribute('noun(257274-2-8)',owned_by('noun(257274-2-a296)'))).
lf(inst('noun(257274-2-8)','F/A-18A')).
lf(attribute('noun(257274-2-6)',owned_by('noun(257274-2-a296)'))).
lf(attribute('noun(257274-2-6)',next_to('noun(257274-2-8)'))).
lf(inst('noun(257274-2-6)','A-6E')).
lf(attribute('noun(257274-2-4)',owned_by('noun(257274-2-a296)'))).
lf(attribute('noun(257274-2-4)',next_to('noun(257274-2-6)'))).
lf(inst('noun(257274-2-4)','A-7E')).
lf(attribute('noun(257274-2-2)',owned_by('noun(257274-2-a297)'))).
lf(attribute('noun(257274-2-2)',next_to('noun(257274-2-4)'))).
lf(quantity('noun(257274-2-2)',all)).
lf(inst('noun(257274-2-2)','A-6B')).
lf(theme('noun(257274-1-1)',obj('noun(257274-1-3)'))).
lf(inst('noun(257274-1-1)','air-to-air view')).
lf(object('pastpart(257274-3-1)',with('noun(257274-3-3)'))).
lf(location('pastpart(257274-3-1)',on('noun(257274-2-2)'))).
lf(activity('pastpart(257274-3-1)',assemble)).
lf(quantity('noun(257274-1-3)',4)).
lf(attribute('noun(257274-1-3)',inst('noun(257274-2-8)'))).
lf(attribute('noun(257274-1-3)',inst('noun(257274-2-6)'))).
lf(attribute('noun(257274-1-3)',inst('noun(257274-2-4)'))).
lf(attribute('noun(257274-1-3)',inst('noun(257274-2-2)'))).
lf(location('noun(257274-1-3)',in('noun(257274-1-a298)'))).
lf(inst('noun(257274-1-3)',formation)).
lf(inst('noun(257274-1-a298)',air)).
lf(inst('noun(257274-2-a296)','NWC')).
::::::::::::::
258795
::::::::::::::

lf(attribute('noun(258795-2-2)',inst('noun(258795-2-a301)'))).
lf(inst('noun(258795-2-2)','Antonio')).
lf(quantity('noun(258795-3-1)',full)).
lf(theme('noun(258795-3-1)',obj('noun(258795-1-3)'))).
lf(inst('noun(258795-3-1)','side view')).
lf(inst('noun(258795-2-a301)','Commander')).
lf(inst('noun(258795-1-a299)','VX-5')).
lf(attribute('noun(258795-1-3)',owned_by('noun(258795-1-a299)'))).
lf(attribute('noun(258795-1-3)',has_part('noun(258795-1-4)'))).
lf(time('noun(258795-1-3)',after('noun(258795-3-3)'))).
lf(inst('noun(258795-1-3)','F/A-18A')).
lf(inst('noun(258795-3-3)',sunset)).
lf(inst('noun(258795-2-a300)','night vision')).
lf(attribute('noun(258795-2-4)',inst('noun(258795-2-a300)'))).
lf(location('noun(258795-2-4)',in('noun(258795-2-5)'))).
lf(inst('noun(258795-2-4)',goggles)).
```

```
lf(object('pastpart(258795-1-1)',with('noun(258795-1-5)'))).
lf(object('pastpart(258795-1-1)',with('noun(258795-1-6)'))).
lf(object('pastpart(258795-1-1)',with('noun(258795-1-8)'))).
lf(location('pastpart(258795-1-1)',on('noun(258795-1-3)'))).
lf(activity('pastpart(258795-1-1)',assemble)).
lf(quantity('noun(258795-1-8)',plural)).
lf(inst('noun(258795-1-8)','Sidewinder')).
lf(inst('noun(258795-1-6)','Maverick')).
lf(inst('noun(258795-1-5)','Harm')).
lf(agent('pverb(258795-2-a302)',obj('noun(258795-2-2)'))).
lf(theme('pverb(258795-2-a302)',obj('noun(258795-2-4)'))).
lf(activity('pverb(258795-2-a302)',wear)).
lf(inst('noun(258795-2-5)',cockpit)).
lf(attribute('noun(258795-1-4)','35')).
lf(inst('noun(258795-1-4)',nose)).
::::::::::::::
262865
::::::::::::::

lf(location('noun(262865-1-4)',on('noun(262865-1-5)'))).
lf(inst('noun(262865-1-4)','AIM-9R')).
lf(quantity('noun(262865-1-1)','3/4')).
lf(theme('noun(262865-1-1)',obj('noun(262865-1-4)'))).
lf(inst('noun(262865-1-1)','front view')).
lf(inst('noun(262865-1-5)',stand)).
::::::::::::::
262866
::::::::::::::

lf(location('noun(262866-1-4)',on('noun(262866-1-5)'))).
lf(inst('noun(262866-1-4)','AIM-9R')).
lf(theme('noun(262866-1-1)',obj('noun(262866-1-4)'))).
lf(inst('noun(262866-1-1)','front view')).
lf(inst('noun(262866-1-5)',stand)).
::::::::::::::
262867
::::::::::::::

lf(location('noun(262867-1-4)',on('noun(262867-1-5)'))).
lf(inst('noun(262867-1-4)','AIM-9R')).
lf(quantity('noun(262867-1-1)','3/4')).
lf(theme('noun(262867-1-1)',obj('noun(262867-1-4)'))).
lf(inst('noun(262867-1-1)','front view')).
lf(inst('noun(262867-1-5)',stand)).
::::::::::::::
262868
::::::::::::::

lf(inst('noun(262868-1-3)','AIM-9R')).
lf(quantity('noun(262868-2-1)',closeup)).
lf(theme('noun(262868-2-1)',obj('noun(262868-1-3)'))).
lf(inst('noun(262868-2-1)','side view')).
lf(attribute('noun(262868-1-6)',has_part('noun(262868-1-7)'))).
lf(attribute('noun(262868-1-6)',inst('noun(262868-1-5)'))).
lf(inst('noun(262868-1-6)','F/A-18C')).
lf(attribute('noun(262868-1-5)','163284')).
lf(inst('noun(262868-1-5)',bureau_no)).
lf(theme('pastpart(262868-1-1)',obj('noun(262868-1-3)'))).
lf(location('pastpart(262868-1-1)',on('noun(262868-1-6)'))).
lf(location('pastpart(262868-1-1)',on('noun(262868-2-4)'))).
lf(activity('pastpart(262868-1-1)',assemble)).
```

```
lf(attribute('noun(262868-2-a303)',outboard)).
lf(inst('noun(262868-2-a303)',wing)).
lf(attribute('noun(262868-2-4)',part_of('noun(262868-2-a303)'))).
lf(inst('noun(262868-2-4)',pylon)).
lf(attribute('noun(262868-1-7)','110')).
lf(inst('noun(262868-1-7)',nose)).
::::::::::::::
262869
::::::::::::::

lf(inst('noun(262869-1-3)','AIM-9R')).
lf(quantity('noun(262869-2-1)',closeup)).
lf(theme('noun(262869-2-1)',obj('noun(262869-1-3)'))).
lf(inst('noun(262869-2-1)','front view')).
lf(attribute('noun(262869-1-6)',has_part('noun(262869-1-7)'))).
lf(attribute('noun(262869-1-6)',inst('noun(262869-1-5)'))).
lf(inst('noun(262869-1-6)','F/A-18C')).
lf(attribute('noun(262869-1-5)','163284')).
lf(inst('noun(262869-1-5)',bureau_no)).
lf(theme('pastpart(262869-1-1)',obj('noun(262869-1-3)'))).
lf(location('pastpart(262869-1-1)',on('noun(262869-1-6)'))).
lf(location('pastpart(262869-1-1)',on('noun(262869-2-4)'))).
lf(activity('pastpart(262869-1-1)',assemble)).
lf(attribute('noun(262869-2-a304)',outboard)).
lf(inst('noun(262869-2-a304)',wing)).
lf(attribute('noun(262869-2-4)',part_of('noun(262869-2-a304)'))).
lf(inst('noun(262869-2-4)',pylon)).
lf(attribute('noun(262869-1-7)','110')).
lf(inst('noun(262869-1-7)',nose)).
::::::::::::::
262870
::::::::::::::

lf(inst('noun(262870-1-3)','AIM-9R')).
lf(attribute('noun(262870-1-6)',has_part('noun(262870-1-7)'))).
lf(attribute('noun(262870-1-6)',inst('noun(262870-1-5)'))).
lf(inst('noun(262870-1-6)','F/A-18C')).
lf(attribute('noun(262870-1-5)','163284')).
lf(inst('noun(262870-1-5)',bureau_no)).
lf(location('noun(262870-2-2)',on('noun(262870-1-3)'))).
lf(location('noun(262870-2-2)',on('noun(262870-2-4)'))).
lf(inst('noun(262870-2-2)',front)).
lf(quantity('noun(262870-2-1)',closeup)).
lf(theme('noun(262870-2-1)',obj('noun(262870-2-2)'))).
lf(inst('noun(262870-2-1)',view)).
lf(theme('pastpart(262870-1-1)',obj('noun(262870-1-3)'))).
lf(location('pastpart(262870-1-1)',on('noun(262870-1-6)'))).
lf(activity('pastpart(262870-1-1)',assemble)).
lf(inst('noun(262870-2-4)',launcher)).
lf(attribute('noun(262870-1-7)','110')).
lf(inst('noun(262870-1-7)',nose)).
::::::::::::::
262871
::::::::::::::

lf(inst('noun(262871-1-3)','AIM-9R')).
lf(quantity('noun(262871-2-1)','3/4')).
lf(theme('noun(262871-2-1)',obj('noun(262871-1-6)'))).
lf(inst('noun(262871-2-1)','front view')).
lf(attribute('noun(262871-1-6)',has_part('noun(262871-1-7)'))).
lf(attribute('noun(262871-1-6)',inst('noun(262871-1-5)'))).
```

```
lf(inst('noun(262871-1-6)','F/A-18C')).
lf(attribute('noun(262871-1-5)','163284')).
lf(inst('noun(262871-1-5)',bureau_no)).
lf(theme('pastpart(262871-1-1)',obj('noun(262871-1-3)'))).
lf(location('pastpart(262871-1-1)',on('noun(262871-1-6)'))).
lf(activity('pastpart(262871-1-1)',assemble)).
lf(attribute('noun(262871-1-7)','110')).
lf(inst('noun(262871-1-7)',nose)).
::::::::::::::
262872
::::::::::::::

lf(inst('noun(262872-1-3)','AIM-9R')).
lf(quantity('noun(262872-2-1)','3/4')).
lf(theme('noun(262872-2-1)',obj('noun(262872-1-6)'))).
lf(inst('noun(262872-2-1)','front view')).
lf(attribute('noun(262872-1-6)',has_part('noun(262872-1-7)'))).
lf(attribute('noun(262872-1-6)',inst('noun(262872-1-5)'))).
lf(inst('noun(262872-1-6)','F/A-18C')).
lf(attribute('noun(262872-1-5)','163284')).
lf(inst('noun(262872-1-5)',bureau_no)).
lf(theme('pastpart(262872-1-1)',obj('noun(262872-1-3)'))).
lf(location('pastpart(262872-1-1)',on('noun(262872-1-6)'))).
lf(activity('pastpart(262872-1-1)',assemble)).
lf(attribute('noun(262872-1-7)','110')).
lf(inst('noun(262872-1-7)',nose)).
::::::::::::::
262873
::::::::::::::

lf(inst('noun(262873-1-3)','AIM-9R')).
lf(quantity('noun(262873-2-1)',full)).
lf(theme('noun(262873-2-1)',obj('noun(262873-1-6)'))).
lf(inst('noun(262873-2-1)','side view')).
lf(attribute('noun(262873-1-6)',has_part('noun(262873-1-7)'))).
lf(attribute('noun(262873-1-6)',inst('noun(262873-1-5)'))).
lf(inst('noun(262873-1-6)','F/A-18C')).
lf(attribute('noun(262873-1-5)','163284')).
lf(inst('noun(262873-1-5)',bureau_no)).
lf(theme('pastpart(262873-1-1)',obj('noun(262873-1-3)'))).
lf(location('pastpart(262873-1-1)',on('noun(262873-1-6)'))).
lf(activity('pastpart(262873-1-1)',assemble)).
lf(attribute('noun(262873-1-7)','110')).
lf(inst('noun(262873-1-7)',nose)).
::::::::::::::
264968
::::::::::::::

lf(inst('noun(264968-1-3)','AIM-9R')).
lf(inst('noun(264968-2-3)','QF-86')).
lf(inst('noun(264968-1-1)','F/A-18')).
lf(agent('prespart(264968-2-1)',obj('noun(264968-1-3)'))).
lf(theme('prespart(264968-2-1)',obj('noun(264968-2-3)'))).
lf(activity('prespart(264968-2-1)',hit)).
lf(theme('prespart(264968-1-1)',obj('noun(264968-1-3)'))).
lf(source('prespart(264968-1-1)',from('noun(264968-1-1)'))).
lf(activity('prespart(264968-1-1)',launch)).
::::::::::::::
29263
::::::::::::::
```

```
lf(inst('noun(29263-2-2)',camera)).
lf(inst('noun(29263-1-4)','Michelson Laboratory')).
lf(theme('pastpart(29263-2-1)',obj('noun(29263-1-1)'))).
lf(object('pastpart(29263-2-1)',with('noun(29263-2-2)'))).
lf(object('pastpart(29263-2-1)',with('noun(29263-2-3)'))).
lf(activity('pastpart(29263-2-1)',assemble)).
lf(inst('noun(29263-1-3)','Machine Shop')).
lf(location('noun(29263-1-1)',at('noun(29263-1-4)'))).
lf(location('noun(29263-1-1)',in('noun(29263-1-3)'))).
lf(inst('noun(29263-1-1)','TIM')).
lf(inst('noun(29263-2-3)',operator)).
::::::::::::::
29293
::::::::::::::

lf(attribute('noun(29293-1-4)',inst('noun(29293-1-5)'))).
lf(attribute('noun(29293-1-4)',inst('noun(29293-1-3)'))).
lf(location('noun(29293-1-4)',in('noun(29293-1-a305)'))).
lf(inst('noun(29293-1-4)','A-4C')).
lf(inst('noun(29293-1-9)','MK-4')).
lf(theme('noun(29293-2-1)',obj('noun(29293-1-9)'))).
lf(attribute('noun(29293-2-1)',excellent)).
lf(inst('noun(29293-2-1)','side view')).
lf(attribute('noun(29293-1-3)','145063')).
lf(inst('noun(29293-1-3)',bureau_no)).
lf(theme('noun(29293-1-1)',obj('noun(29293-1-4)'))).
lf(inst('noun(29293-1-1)','air-to-air view')).
lf(location('noun(29293-1-5)',on('noun(29293-1-6)'))).
lf(inst('noun(29293-1-5)','China Lake')).
lf(inst('noun(29293-1-a305)',air)).
lf(agent('prespart(29293-1-1)',obj('noun(29293-1-4)'))).
lf(theme('prespart(29293-1-1)',obj('noun(29293-1-9)'))).
lf(activity('prespart(29293-1-1)',launch)).
lf(inst('noun(29293-1-6)',tail)).
::::::::::::::
29773
::::::::::::::

lf(location('noun(29773-2-7)',on('noun(29773-3-2)'))).
lf(inst('noun(29773-2-7)','AIM-9C')).
lf(quantity('noun(29773-1-3)',plural)).
lf(inst('noun(29773-1-3)','AIM-9D')).
lf(inst('noun(29773-1-1)','AIM-9B')).
lf(theme('noun(29773-3-1)',obj('noun(29773-2-7)'))).
lf(attribute('noun(29773-3-1)',right)).
lf(quantity('noun(29773-3-1)','3/4')).
lf(inst('noun(29773-3-1)','side view')).
lf(theme('noun(29773-2-1)',obj('noun(29773-1-1)'))).
lf(theme('noun(29773-2-1)',obj('noun(29773-1-3)'))).
lf(theme('noun(29773-2-1)',obj('noun(29773-2-7)'))).
lf(inst('noun(29773-2-1)',comparison)).
lf(inst('noun(29773-3-2)',stand)).
::::::::::::::
3204
::::::::::::::

lf(quantity('noun(3204-2-2)',3)).
lf(location('noun(3204-2-2)',on('noun(3204-2-3)'))).
lf(quantity('noun(3204-2-2)',2)).
lf(inst('noun(3204-2-2)','F-3D-1')).
lf(theme('pres(3204-3-1)',obj('noun(3204-3-3)'))).
```

```
lf(location('pres(3204-3-1)',on('noun(3204-2-2)'))).
lf(activity('pres(3204-3-1)',assemble)).
lf(quantity('noun(3204-3-3)',plural)).
lf(inst('noun(3204-3-3)','Sidewinder')).
lf(inst('noun(3204-2-3)','flight path')).
lf(attribute('noun(3204-1-1)','1424')).
lf(inst('noun(3204-1-1)','test plan')).
::::::::::::::
34070
::::::::::::::

lf(attribute('noun(34070-1-4)',inst('noun(34070-1-5)'))).
lf(attribute('noun(34070-1-4)',inst('noun(34070-1-3)'))).
lf(location('noun(34070-1-4)',in('noun(34070-1-a307)'))).
lf(inst('noun(34070-1-4)','A-4B')).
lf(source('noun(34070-1-7)',from('noun(34070-3-3)'))).
lf(location('noun(34070-1-7)',on('noun(34070-1-4)'))).
lf(location('noun(34070-1-7)',in('noun(34070-1-a307)'))).
lf(inst('noun(34070-1-7)','Snipe')).
lf(attribute('noun(34070-1-3)','142777')).
lf(inst('noun(34070-1-3)',bureau_no)).
lf(theme('noun(34070-1-1)',obj('noun(34070-1-4)'))).
lf(theme('noun(34070-1-1)',obj('noun(34070-1-7)'))).
lf(inst('noun(34070-1-1)','air-to-air view')).
lf(location('noun(34070-3-3)',on('noun(34070-1-4)'))).
lf(inst('noun(34070-3-3)',underside)).
lf(quantity('noun(34070-2-a306)',plural)).
lf(inst('noun(34070-2-a306)',bomb)).
lf(attribute('noun(34070-2-4)',holds('noun(34070-2-a306)'))).
lf(attribute('noun(34070-2-4)',part_of('noun(34070-2-2)'))).
lf(inst('noun(34070-2-4)',rack)).
lf(inst('noun(34070-2-2)','centerline station')).
lf(attribute('noun(34070-2-1)','3')).
lf(location('noun(34070-2-1)',on('noun(34070-2-2)'))).
lf(inst('noun(34070-2-1)',round)).
lf(inst('noun(34070-1-5)','China Lake')).
lf(inst('noun(34070-1-a307)',air)).
::::::::::::::
34271
::::::::::::::

lf(inst('noun(34271-1-6)','A-4E')).
lf(quantity('noun(34271-1-2)',plural)).
lf(source('noun(34271-1-2)',from('noun(34271-1-6)'))).
lf(time('noun(34271-1-2)',after('noun(34271-1-4)'))).
lf(inst('noun(34271-1-2)','CLAM B')).
lf(location('noun(34271-2-3)',on('noun(34271-2-4)'))).
lf(inst('noun(34271-2-3)',center)).
lf(attribute('noun(34271-2-1)',excellent)).
lf(theme('noun(34271-2-1)',obj('noun(34271-2-2)'))).
lf(inst('noun(34271-2-1)',view)).
lf(theme('noun(34271-1-4)',obj('noun(34271-1-2)'))).
lf(activity('noun(34271-1-4)',launch)).
lf(state('noun(34271-2-4)',depart)).
lf(inst('noun(34271-2-4)',fireball)).
lf(location('noun(34271-2-2)',in('noun(34271-2-3)'))).
lf(inst('noun(34271-2-2)',target)).
::::::::::::::
38239
::::::::::::::
```

```
lf(inst('noun(38239-2-3)',trailer)).
lf(attribute('noun(38239-1-1)',photograph)).
lf(location('noun(38239-1-1)',on('noun(38239-2-a308)'))).
lf(location('noun(38239-1-1)',on('noun(38239-2-3)'))).
lf(inst('noun(38239-1-1)','tracking mount')).
lf(state('noun(38239-2-a308)',extend)).
lf(inst('noun(38239-2-a308)',range)).
::::::::::::::
40226
::::::::::::::

lf(location('noun(40226-1-2)',on('noun(40226-1-4)'))).
lf(location('noun(40226-1-2)',at('noun(40226-2-3)'))).
lf(inst('noun(40226-1-2)','Adam')).
lf(inst('noun(40226-1-4)','S-2A')).
lf(attribute('noun(40226-2-3)',down)).
lf(quantity('noun(40226-2-3)',degree('17'))).
lf(inst('noun(40226-2-3)',tilt)).
::::::::::::::
41136
::::::::::::::

lf(inst('noun(41136-1-6)','UH-1E')).
lf(attribute('noun(41136-1-4)',inst('noun(41136-1-a309)'))).
lf(source('noun(41136-1-4)',from('noun(41136-1-6)'))).
lf(inst('noun(41136-1-4)',sequence)).
lf(quantity('noun(41136-1-2)',plural(4))).
lf(inst('noun(41136-1-2)','HTW')).
lf(quantity('noun(41136-1-a309)',plural)).
lf(theme('noun(41136-1-a309)',obj('noun(41136-1-2)'))).
lf(activity('noun(41136-1-a309)',launch)).
lf(theme('pastpart(41136-2-1)',obj('noun(41136-2-1)'))).
lf(location('pastpart(41136-2-1)',on('noun(41136-1-2)'))).
lf(activity('pastpart(41136-2-1)',open_act)).
lf(quantity('noun(41136-2-1)',plural)).
lf(inst('noun(41136-2-1)',parachute)).
::::::::::::::
43176
::::::::::::::

lf(attribute('noun(43176-1-6)',inst('noun(43176-1-7)'))).
lf(attribute('noun(43176-1-6)',inst('noun(43176-1-5)'))).
lf(inst('noun(43176-1-6)','A-4C')).
lf(quantity('noun(43176-2-1)',full)).
lf(theme('noun(43176-2-1)',obj('noun(43176-1-6)'))).
lf(inst('noun(43176-2-1)','side view')).
lf(attribute('noun(43176-1-5)','147781')).
lf(inst('noun(43176-1-5)',bureau_no)).
lf(theme('noun(43176-1-1)',obj('noun(43176-1-3)'))).
lf(inst('noun(43176-1-1)','air-to-air view')).
lf(inst('noun(43176-2-a310)','B-1-B')).
lf(attribute('noun(43176-2-7)',part_of('noun(43176-2-a310)'))).
lf(inst('noun(43176-2-7)',center)).
lf(location('noun(43176-1-7)',on('noun(43176-1-8)'))).
lf(inst('noun(43176-1-7)','China Lake')).
lf(location('noun(43176-1-3)',on('noun(43176-1-6)'))).
lf(attribute('noun(43176-1-3)',isa('noun(43176-2-a311)'))).
lf(location('noun(43176-1-3)',over('noun(43176-2-7)'))).
lf(location('noun(43176-1-3)',in('noun(43176-1-a312)'))).
lf(inst('noun(43176-1-3)','Walleye')).
lf(inst('noun(43176-1-a312)',air)).
```

```
lf(inst('noun(43176-2-a311)',dummy)).
lf(inst('noun(43176-1-8)',tail)).
:::::::::::::
43694
:::::::::::::

lf(quantity('noun(43694-1-2)',plural)).
lf(location('noun(43694-1-2)',on('noun(43694-1-4)'))).
lf(inst('noun(43694-1-2)','Rockeye II')).
lf(attribute('noun(43694-1-4)',has_part('noun(43694-1-5)'))).
lf(attribute('noun(43694-1-4)',inst('noun(43694-1-6)'))).
lf(inst('noun(43694-1-4)','A-6A')).
lf(theme('pastpart(43694-2-1)',obj('noun(43694-1-2)'))).
lf(location('pastpart(43694-2-1)',on('noun(43694-1-7)'))).
lf(location('pastpart(43694-2-1)',on('noun(43694-2-3)'))).
lf(location('pastpart(43694-2-1)',on('noun(43694-1-4)'))).
lf(activity('pastpart(43694-2-1)',assemble)).
lf(inst('noun(43694-2-3)','centerline station')).
lf(state('noun(43694-1-8)',fold)).
lf(inst('noun(43694-1-8)',position)).
lf(quantity('noun(43694-1-7)',plural)).
lf(location('noun(43694-1-7)',in('noun(43694-1-8)'))).
lf(attribute('noun(43694-1-7)',part_of('noun(43694-1-4)'))).
lf(inst('noun(43694-1-7)',wing)).
lf(quantity('noun(43694-1-6)',plural(other))).
lf(quantity('noun(43694-1-6)',no)).
lf(inst('noun(43694-1-6)',marking)).
lf(attribute('noun(43694-1-5)','17')).
lf(inst('noun(43694-1-5)',nose)).
:::::::::::::
44263
:::::::::::::

lf(agent('prespart(44263-2-1)',obj('noun(44263-1-1)'))).
lf(activity('prespart(44263-2-1)',approach)).
lf(quantity('noun(44263-2-1)','3/4')).
lf(theme('noun(44263-2-1)',obj('noun(44263-1-1)'))).
lf(inst('noun(44263-2-1)',view)).
lf(inst('noun(44263-1-a313)','Bullpup')).
lf(inst('noun(44263-1-1)','Shrike')).
lf(agent('prespart(44263-2-2)',obj('noun(44263-1-1)'))).
lf(theme('prespart(44263-2-2)',obj('noun(44263-1-5)'))).
lf(activity('prespart(44263-2-2)',injure)).
lf(agent('prespart(44263-1-1)',obj('noun(44263-1-1)'))).
lf(destination('prespart(44263-1-1)',at('noun(44263-1-5)'))).
lf(activity('prespart(44263-1-1)',launch)).
lf(attribute('noun(44263-1-5)',isa('noun(44263-1-a313)'))).
lf(attribute('noun(44263-1-5)',isa('noun(44263-1-1)'))).
lf(attribute('noun(44263-1-5)','SCR-584')).
lf(inst('noun(44263-1-5)',target)).
:::::::::::::
45935
:::::::::::::

lf(quantity('noun(45935-1-1)',plural)).
lf(attribute('noun(45935-1-1)',local)).
lf(inst('noun(45935-1-1)','Beavertail cactus')).
lf(theme('noun(45935-2-2)',obj('noun(45935-1-1)'))).
lf(inst('noun(45935-2-2)',bloom)).
:::::::::::::
45936
```

```
::::::::::::::

lf(quantity('noun(45936-1-1)',plural)).
lf(attribute('noun(45936-1-1)',local)).
lf(inst('noun(45936-1-1)','Dalea')).
lf(theme('noun(45936-2-2)',obj('noun(45936-1-1)'))).
lf(inst('noun(45936-2-2)',bloom)).
::::::::::::::
45937
::::::::::::::

lf(quantity('noun(45937-1-1)',plural)).
lf(attribute('noun(45937-1-1)',local)).
lf(inst('noun(45937-1-1)','Beavertail cactus')).
lf(quantity('noun(45937-2-1)',closeup)).
lf(theme('noun(45937-2-1)',obj('noun(45937-1-1)'))).
lf(inst('noun(45937-2-1)',view)).
::::::::::::::
5824
::::::::::::::

lf(attribute('noun(5824-1-4)',inst('noun(5824-1-5)'))).
lf(attribute('noun(5824-1-4)',inst('noun(5824-1-3)'))).
lf(location('noun(5824-1-4)',in('noun(5824-1-a314)'))).
lf(inst('noun(5824-1-4)','F-3H-1')).
lf(theme('noun(5824-2-1)',obj('noun(5824-1-4)'))).
lf(quantity('noun(5824-2-1)',full)).
lf(theme('noun(5824-2-1)',obj('noun(5824-2-3)'))).
lf(inst('noun(5824-2-1)','side view')).
lf(attribute('noun(5824-1-3)','133550')).
lf(inst('noun(5824-1-3)',bureau_no)).
lf(theme('noun(5824-1-1)',obj('noun(5824-1-4)'))).
lf(inst('noun(5824-1-1)','air-to-air view')).
lf(location('noun(5824-2-3)',on('noun(5824-1-4)'))).
lf(inst('noun(5824-2-3)','Sidewinder')).
lf(location('noun(5824-1-5)',on('noun(5824-1-6)'))).
lf(inst('noun(5824-1-5)','China Lake')).
lf(inst('noun(5824-1-a314)',air)).
lf(inst('noun(5824-1-6)',tail)).
::::::::::::::
62439
::::::::::::::

lf(theme('prespart(62439-3-1)',obj('noun(62439-3-3)'))).
lf(agent('prespart(62439-3-1)',obj('noun(62439-3-2)'))).
lf(activity('prespart(62439-3-1)',show)).
lf(theme('noun(62439-3-3)',obj('noun(62439-3-2)'))).
lf(force('noun(62439-3-3)',obj('noun(62439-3-4)'))).
lf(inst('noun(62439-3-3)',erosion)).
lf(inst('noun(62439-2-a315)','Kern River')).
lf(theme('noun(62439-2-2)',obj('noun(62439-2-a315)'))).
lf(quantity('noun(62439-2-2)',plural(closeup))).
lf(theme('noun(62439-2-2)',obj('noun(62439-3-2)'))).
lf(inst('noun(62439-2-2)',view)).
lf(inst('noun(62439-3-4)',water)).
lf(quantity('noun(62439-3-2)',plural)).
lf(attribute('noun(62439-3-2)',large)).
lf(inst('noun(62439-3-2)',rock)).
::::::::::::::
62440
::::::::::::::
```

```
lf(attribute('noun(62440-4-1)',small)).
lf(location('noun(62440-4-1)',on('noun(62440-4-2)'))).
lf(inst('noun(62440-4-1)',girl)).
lf(attribute('noun(62440-2-a316)',down)).
lf(inst('noun(62440-2-a316)','Kern River')).
lf(quantity('noun(62440-2-2)',plural)).
lf(theme('noun(62440-2-2)',obj('noun(62440-2-a316)'))).
lf(inst('noun(62440-2-2)',view)).
lf(inst('noun(62440-4-2)',rock)).
lf(theme('prespart(62440-3-1)',obj('noun(62440-2-a316)'))).
lf(activity('prespart(62440-3-1)',observe)).
::::::::::::::
62441
::::::::::::::

lf(inst('noun(62441-4-4)',background)).
lf(quantity('noun(62441-4-1)',plural)).
lf(inst('noun(62441-4-1)',tree)).
lf(inst('noun(62441-3-2)',foreground)).
lf(location('noun(62441-2-a317)',in('noun(62441-3-2)'))).
lf(inst('noun(62441-2-a317)','Kern River')).
lf(quantity('noun(62441-2-2)',plural)).
lf(theme('noun(62441-2-2)',obj('noun(62441-2-a317)'))).
lf(inst('noun(62441-2-2)',view)).
lf(inst('noun(62441-4-a318)',mountain)).
lf(attribute('noun(62441-4-3)',inst('noun(62441-4-a318)'))).
lf(location('noun(62441-4-3)',in('noun(62441-4-4)'))).
lf(inst('noun(62441-4-3)',range)).
::::::::::::::
64287
::::::::::::::

lf(inst('noun(64287-3-2)','F-14A')).
lf(location('noun(64287-1-3)',in('noun(64287-1-a319)'))).
lf(inst('noun(64287-1-3)','Phoenix')).
lf(theme('noun(64287-1-1)',obj('noun(64287-1-3)'))).
lf(inst('noun(64287-1-1)','air-to-air view')).
lf(theme('noun(64287-2-1)',obj('noun(64287-1-5)'))).
lf(attribute('noun(64287-2-1)',large)).
lf(inst('noun(64287-2-1)',explosion)).
lf(inst('noun(64287-1-5)','QF-9')).
lf(inst('noun(64287-1-a319)',air)).
lf(theme('prespart(64287-3-1)',obj('noun(64287-1-3)'))).
lf(source('prespart(64287-3-1)',from('noun(64287-3-2)'))).
lf(activity('prespart(64287-3-1)',launch)).
lf(agent('prespart(64287-1-1)',obj('noun(64287-1-3)'))).
lf(theme('prespart(64287-1-1)',obj('noun(64287-1-5)'))).
lf(location('prespart(64287-1-1)',in('noun(64287-1-1)'))).
lf(activity('prespart(64287-1-1)',hit)).
::::::::::::::
64288
::::::::::::::

lf(inst('noun(64288-3-2)','F-14A')).
lf(location('noun(64288-1-3)',in('noun(64288-1-a320)'))).
lf(inst('noun(64288-1-3)','Phoenix')).
lf(theme('noun(64288-1-1)',obj('noun(64288-1-3)'))).
lf(inst('noun(64288-1-1)','air-to-air view')).
lf(theme('noun(64288-2-1)',obj('noun(64288-1-5)'))).
lf(attribute('noun(64288-2-1)',large)).
```

```
lf(inst('noun(64288-2-1)',explosion)).
lf(inst('noun(64288-1-5)','QF-9')).
lf(inst('noun(64288-1-a320)',air)).
lf(inst('noun(64288-4-2)',engine)).
lf(theme('prespart(64288-3-1)',obj('noun(64288-1-3)'))).
lf(theme('prespart(64288-3-1)',obj('noun(64288-4-2)'))).
lf(location('prespart(64288-3-1)',under('noun(64288-1-5)'))).
lf(source('prespart(64288-3-1)',from('noun(64288-3-2)'))).
lf(activity('prespart(64288-3-1)',launch)).
lf(agent('prespart(64288-1-1)',obj('noun(64288-1-3)'))).
lf(theme('prespart(64288-1-1)',obj('noun(64288-1-5)'))).
lf(location('prespart(64288-1-1)',in('noun(64288-1-1)'))).
lf(activity('prespart(64288-1-1)',hit)).
::::::::::::::
64289
::::::::::::::

lf(inst('noun(64289-3-2)','F-14A')).
lf(location('noun(64289-1-3)',in('noun(64289-1-a15)'))).
lf(inst('noun(64289-1-3)','Phoenix')).
lf(theme('noun(64289-1-1)',obj('noun(64289-1-3)'))).
lf(inst('noun(64289-1-1)','air-to-air view')).
lf(activity('noun(64289-4-2)',fly)).
lf(theme('noun(64289-2-1)',obj('noun(64289-1-5)'))).
lf(attribute('noun(64289-2-1)',large)).
lf(inst('noun(64289-2-1)',explosion)).
lf(inst('noun(64289-1-5)','QF-9')).
lf(inst('noun(64289-1-a15)',air)).
lf(location('noun(64289-6-1)',behind('noun(64289-1-5)'))).
lf(inst('noun(64289-6-1)',plume)).
lf(agent('prespart(64289-5-1)',obj('noun(64289-5-1)'))).
lf(activity('prespart(64289-5-1)',heat)).
lf(inst('noun(64289-5-1)',engine)).
lf(theme('prespart(64289-3-1)',obj('noun(64289-1-3)'))).
lf(location('prespart(64289-3-1)',in('noun(64289-4-2)'))).
lf(source('prespart(64289-3-1)',from('noun(64289-3-2)'))).
lf(activity('prespart(64289-3-1)',launch)).
lf(agent('prespart(64289-1-1)',obj('noun(64289-1-3)'))).
lf(theme('prespart(64289-1-1)',obj('noun(64289-1-5)'))).
lf(location('prespart(64289-1-1)',in('noun(64289-1-1)'))).
lf(activity('prespart(64289-1-1)',hit)).
::::::::::::::
64472
::::::::::::::

lf(inst('noun(64472-3-2)',crane)).
lf(state('noun(64472-2-a322)',direct)).
lf(inst('noun(64472-2-a322)',intensity)).
lf(quantity('noun(64472-2-2)',plural)).
lf(attribute('noun(64472-2-2)',inst('noun(64472-2-a322)'))).
lf(attribute('noun(64472-2-2)',solar)).
lf(inst('noun(64472-2-2)',study)).
lf(inst('noun(64472-1-1)',heliostat)).
lf(theme('pastpart(64472-3-1)',obj('noun(64472-3-1)'))).
lf(instrument('pastpart(64472-3-1)',with('noun(64472-3-2)'))).
lf(activity('pastpart(64472-3-1)',ptrans)).
lf(inst('noun(64472-3-1)',target)).
::::::::::::::
64473
::::::::::::::
```

```
lf(theme('noun(64473-3-2)',obj('noun(64473-3-1)'))).
lf(inst('noun(64473-3-2)',hoisting)).
lf(state('noun(64473-2-a323)',direct)).
lf(inst('noun(64473-2-a323)',intensity)).
lf(quantity('noun(64473-2-2)',plural)).
lf(attribute('noun(64473-2-2)',inst('noun(64473-2-a323)'))).
lf(attribute('noun(64473-2-2)',solar)).
lf(inst('noun(64473-2-2)',study)).
lf(inst('noun(64473-1-1)',heliostat)).
lf(theme('prespart(64473-3-1)',obj('noun(64473-3-1)'))).
lf(reason('prespart(64473-3-1)',for('noun(64473-3-2)'))).
lf(activity('prespart(64473-3-1)',prepare)).
lf(inst('noun(64473-3-1)',target)).
::::::::::::::
64474
::::::::::::::

lf(state('noun(64474-2-a324)',direct)).
lf(inst('noun(64474-2-a324)',intensity)).
lf(quantity('noun(64474-2-2)',plural)).
lf(attribute('noun(64474-2-2)',inst('noun(64474-2-a324)'))).
lf(attribute('noun(64474-2-2)',solar)).
lf(inst('noun(64474-2-2)',study)).
lf(inst('noun(64474-1-1)',heliostat)).
lf(location('noun(64474-3-2)',under('noun(64474-3-1)'))).
lf(inst('noun(64474-3-2)',tower)).
lf(inst('noun(64474-3-1)',target)).
::::::::::::::
65019
::::::::::::::

lf(inst('noun(65019-1-3)','Coso Range')).
lf(attribute('noun(65019-1-1)',wild)).
lf(location('noun(65019-1-1)',on('noun(65019-1-3)'))).
lf(location('noun(65019-1-1)',in('noun(65019-1-2)'))).
lf(quantity('noun(65019-1-1)',1)).
lf(attribute('noun(65019-1-1)',inst('noun(65019-2-2)'))).
lf(inst('noun(65019-1-1)',horse)).
lf(inst('noun(65019-2-3)',camera)).
lf(inst('noun(65019-2-2)',blaze)).
lf(inst('noun(65019-1-2)',snow)).
lf(theme('prespart(65019-2-1)',obj('noun(65019-2-3)'))).
lf(agent('prespart(65019-2-1)',obj('noun(65019-1-1)'))).
lf(activity('prespart(65019-2-1)',observe)).
::::::::::::::
66694
::::::::::::::

lf(attribute('noun(66694-2-4)',inst('noun(66694-2-a325)'))).
lf(inst('noun(66694-2-4)','flight path')).
lf(theme('noun(66694-2-2)',obj('noun(66694-2-4)'))).
lf(theme('noun(66694-2-2)',obj('noun(66694-2-1)'))).
lf(inst('noun(66694-2-2)',view)).
lf(attribute('noun(66694-2-1)','1')).
lf(inst('noun(66694-2-1)',hangar)).
lf(inst('noun(66694-1-2)','Armitage Field')).
lf(inst('noun(66694-2-a325)',aircraft)).
::::::::::::::
66695
::::::::::::::
```

```
lf(theme('noun(66695-2-1)',obj('noun(66695-2-2)'))).
lf(theme('noun(66695-2-1)',obj('noun(66695-2-3)'))).
lf(inst('noun(66695-2-1)',overview)).
lf(inst('noun(66695-2-3)','Hot Line')).
lf(inst('noun(66695-2-2)','Cold Line')).
lf(inst('noun(66695-1-2)','Armitage Field')).
::::::::::::::
66696
::::::::::::::

lf(attribute('noun(66696-2-2)','3')).
lf(inst('noun(66696-2-2)',hangar)).
lf(location('noun(66696-2-1)',on('noun(66696-2-2)'))).
lf(inst('noun(66696-2-1)',top)).
lf(inst('noun(66696-1-2)','Armitage Field')).
lf(direction('prespart(66696-2-1)','NW')).
lf(source('prespart(66696-2-1)',from('noun(66696-2-1)'))).
lf(activity('prespart(66696-2-1)',observe)).
::::::::::::::
69812
::::::::::::::

lf(inst('noun(69812-2-a328)','Richard')).
lf(attribute('noun(69812-2-2)',inst('noun(69812-2-a328)'))).
lf(attribute('noun(69812-2-2)',employee_of('noun(69812-2-a329)'))).
lf(attribute('noun(69812-2-2)',next_to('noun(69812-2-5)'))).
lf(attribute('noun(69812-2-2)',next_to('noun(69812-2-7)'))).
lf(attribute('noun(69812-2-2)',next_to('noun(69812-2-2)'))).
lf(inst('noun(69812-2-2)','Fulmer')).
lf(attribute('noun(69812-1-a326)',part_of('noun(69812-1-2)'))).
lf(inst('noun(69812-1-a326)','photovoltaic cell')).
lf(inst('noun(69812-2-a329)','NASA')).
lf(quantity('noun(69812-2-5)',plural)).
lf(inst('noun(69812-2-5)','electric battery')).
lf(inst('noun(69812-2-a327)',power)).
lf(attribute('noun(69812-2-7)',inst('noun(69812-2-a327)'))).
lf(inst('noun(69812-2-7)',inverter)).
lf(theme('infin(69812-1-1)',obj('noun(69812-1-4)'))).
lf(attribute('infin(69812-1-1)',ultimately)).
lf(activity('infin(69812-1-1)',perform)).
lf(theme('prespart(69812-1-1)',obj('noun(69812-1-3)'))).
lf(goal('prespart(69812-1-1)',activity('infin(69812-1-1)'))).
lf(agent('prespart(69812-1-1)',obj('noun(69812-1-2)'))).
lf(activity('prespart(69812-1-1)',pbuild)).
lf(inst('noun(69812-1-4)',radar)).
lf(inst('noun(69812-1-3)',power)).
lf(quantity('noun(69812-1-2)',plural)).
lf(inst('noun(69812-1-2)',panel)).
::::::::::::::
85486
::::::::::::::

lf(inst('noun(85486-2-5)','G-1 Range')).
lf(inst('noun(85486-2-8)','QF-9F')).
lf(inst('noun(85486-4-3)',background)).
lf(theme('prespart(85486-3-1)',obj('noun(85486-3-3)'))).
lf(activity('prespart(85486-3-1)',show)).
lf(theme('noun(85486-3-1)',obj('noun(85486-3-3)'))).
lf(inst('noun(85486-3-1)',view)).
lf(inst('noun(85486-2-a8)','RAPEC')).
lf(attribute('noun(85486-2-a6)',part_of('noun(85486-2-5)'))).
```

```
lf(inst('noun(85486-2-a6)','T-5')).
lf(theme('noun(85486-2-3)',obj('noun(85486-2-a7)'))).
lf(attribute('noun(85486-2-3)',related_program('noun(85486-2-a8)'))).
lf(source('noun(85486-2-3)',from('noun(85486-2-8)'))).
lf(location('noun(85486-2-3)',over('noun(85486-2-5)'))).
lf(inst('noun(85486-2-3)',ejection)).
lf(theme('noun(85486-3-3)',obj('noun(85486-2-a7)'))).
lf(activity('noun(85486-3-3)',launch)).
lf(inst('noun(85486-4-a9)',ship)).
lf(attribute('noun(85486-4-2)','#E')).
lf(attribute('noun(85486-4-2)',isa('noun(85486-4-a9)'))).
lf(location('noun(85486-4-2)',in('noun(85486-4-3)'))).
lf(inst('noun(85486-4-2)',target)).
lf(inst('noun(85486-2-a7)',seat)).
lf(attribute('noun(85486-1-1)','163')).
lf(inst('noun(85486-1-1)',project)).
::::::::::::::
85487
::::::::::::::

lf(inst('noun(85487-2-5)','G-1 Range')).
lf(inst('noun(85487-2-8)','QF-9F')).
lf(theme('prespart(85487-3-1)',obj('noun(85487-3-2)'))).
lf(activity('prespart(85487-3-1)',show)).
lf(theme('noun(85487-3-1)',obj('noun(85487-3-2)'))).
lf(inst('noun(85487-3-1)',view)).
lf(inst('noun(85487-2-a12)','RAPEC')).
lf(attribute('noun(85487-2-a10)',part_of('noun(85487-2-5)'))).
lf(inst('noun(85487-2-a10)','T-5')).
lf(theme('noun(85487-2-3)',obj('noun(85487-2-a11)'))).
lf(attribute('noun(85487-2-3)',related_program('noun(85487-2-a12)'))).
lf(source('noun(85487-2-3)',from('noun(85487-2-8)'))).
lf(location('noun(85487-2-3)',over('noun(85487-2-5)'))).
lf(inst('noun(85487-2-3)',ejection)).
lf(inst('noun(85487-4-a13)',ship)).
lf(inst('noun(85487-4-3)',land)).
lf(attribute('noun(85487-4-2)','#E')).
lf(attribute('noun(85487-4-2)',isa('noun(85487-4-a13)'))).
lf(location('noun(85487-4-2)',under('noun(85487-2-8)'))).
lf(location('noun(85487-4-2)',on('noun(85487-4-3)'))).
lf(inst('noun(85487-4-2)',target)).
lf(theme('prespart(85487-3-2)',obj('noun(85487-3-2)'))).
lf(object('prespart(85487-3-2)',obj('noun(85487-2-8)'))).
lf(activity('prespart(85487-3-2)',free)).
lf(inst('noun(85487-3-2)',dummy)).
lf(inst('noun(85487-2-a11)',seat)).
lf(attribute('noun(85487-1-1)','163')).
lf(inst('noun(85487-1-1)',project)).
::::::::::::::
85488
::::::::::::::

lf(inst('noun(85488-2-5)','G-1 Range')).
lf(attribute('noun(85488-3-3)',inst('noun(85488-3-2)'))).
lf(inst('noun(85488-3-3)','rocket burn')).
lf(inst('noun(85488-2-8)','QF-9F')).
lf(theme('prespart(85488-3-1)',obj('noun(85488-3-2)'))).
lf(activity('prespart(85488-3-1)',show)).
lf(inst('noun(85488-3-2)',end)).
lf(theme('noun(85488-3-1)',obj('noun(85488-3-2)'))).
lf(inst('noun(85488-3-1)',view)).
```

```
lf(inst('noun(85488-2-a16)','RAPEC')).
lf(attribute('noun(85488-2-a14)',part_of('noun(85488-2-5)'))).
lf(inst('noun(85488-2-a14)','T-5')).
lf(theme('noun(85488-2-3)',obj('noun(85488-2-a15)'))).
lf(attribute('noun(85488-2-3)',related_program('noun(85488-2-a16)'))).
lf(source('noun(85488-2-3)',from('noun(85488-2-8)'))).
lf(location('noun(85488-2-3)',over('noun(85488-2-5)'))).
lf(inst('noun(85488-2-3)',ejection)).
lf(inst('noun(85488-2-a15)',seat)).
lf(attribute('noun(85488-1-1)','163')).
lf(inst('noun(85488-1-1)',project)).
::::::::::::::
85489
::::::::::::::

lf(inst('noun(85489-3-4)',pilot)).
lf(inst('noun(85489-2-5)','G-1 Range')).
lf(inst('noun(85489-2-8)','QF-9F')).
lf(theme('prespart(85489-3-1)',obj('noun(85489-3-3)'))).
lf(activity('prespart(85489-3-1)',show)).
lf(agent('noun(85489-3-3)',obj('noun(85489-3-a20)'))).
lf(theme('noun(85489-3-3)',obj('noun(85489-3-4)'))).
lf(theme('noun(85489-3-3)',obj('noun(85489-2-a18)'))).
lf(inst('noun(85489-3-3)',spread)).
lf(theme('noun(85489-3-1)',obj('noun(85489-3-3)'))).
lf(inst('noun(85489-3-1)',view)).
lf(inst('noun(85489-2-a19)','RAPEC')).
lf(attribute('noun(85489-2-a17)',part_of('noun(85489-2-5)'))).
lf(inst('noun(85489-2-a17)','T-5')).
lf(theme('noun(85489-2-3)',obj('noun(85489-2-a18)'))).
lf(attribute('noun(85489-2-3)',related_program('noun(85489-2-a19)'))).
lf(source('noun(85489-2-3)',from('noun(85489-2-8)'))).
lf(location('noun(85489-2-3)',over('noun(85489-2-5)'))).
lf(inst('noun(85489-2-3)',ejection)).
lf(inst('noun(85489-3-a20)',parachute)).
lf(inst('noun(85489-2-a18)',seat)).
lf(attribute('noun(85489-1-1)','163')).
lf(inst('noun(85489-1-1)',project)).
::::::::::::::
900304
::::::::::::::

lf(attribute('noun(900304-1-4)',owned_by('noun(900304-1-a5)'))).
lf(location('noun(900304-1-4)',in('noun(900304-1-a6)'))).
lf(inst('noun(900304-1-4)','B-2')).
lf(theme('noun(900304-1-1)',obj('noun(900304-1-4)'))).
lf(inst('noun(900304-1-1)','air-to-air view')).
lf(attribute('noun(900304-2-1)',(public))).
lf(attribute('noun(900304-2-1)',1)).
lf(time('noun(900304-2-1)',late('1989'))).
lf(agent('noun(900304-2-1)',obj('noun(900304-1-4)'))).
lf(activity('noun(900304-2-1)',fly)).
lf(inst('noun(900304-1-a5)','Northrop')).
lf(inst('noun(900304-1-a6)',air)).
```

APPENDIX G

EMPIRICAL STUDIES

This appendix lists various empirical analysis results for describing the performance of MARIE. Section G.1 lists the NL processing times for each caption in seconds. Performance measures for retrieval effectiveness are based on *recall*, *fallout rate*, and *precision* . The definitions for these measures are repeated here from Chapter VI.

```
recall = number of captions judged good by MARIE & good by Photo Lab
                  number of captions judged good by Photo Lab

fallout = number of captions judged good by MARIE & bad by Photo Lab
                  number of captions judged good by MARIE

precision = 1.0 - fallout
```

We will assume that "good" is the total number of records with the highest score. We also assume that caption scores within 0.5 of the maximum score for a caption identifier from the list of caption identifiers and score results are also considered to be good.

Test queries were obtained from NAWCWPNS China Lake Photo Lab personnel. The procedure for generating the queries are as follows. The 217 captions were shown to the Photo Lab personnel and they were asked to formulate NL queries that were typical of customer queries. From the queries, they were then asked to formulate the appropriate keyphrase queries. Section G.2 lists the Photo Lab queries and search results using the keyphrase approach. Section G.3 lists MARIE's performance using the NL queries themselves. To further test the system, we introduced our own queries, the results of which are shown in Section G.4.

1. NL PROCESSING TIMES PER CAPTION

Table G.1 lists for each caption, the number of words in the caption and the NL processing times. The mean, median, and standard deviation for the number of words per caption and the parsing times per caption are as follows:

	Words	Parse Times
Mean	20.700	17.874
Median	19	15.417
Std. Dev.	9.053	9.753

TABLE G.1. NL PROCESSING TIMES.

Cap	Wd	Sec	Cap	Wd	Sec	Cap	Wd	Sec	Cap	Wd	Sec
900304	19	14.150	124	14	8.733	10851	17	11.633	10862	38	26.683
10880	15	13.467	110169	34	27.800	85486	22	21.200	85487	25	22.667
85488	19	16.084	85489	21	20.283	161044	10	12.200	161045	16	15.217
161082	19	18.100	163030	25	28.467	164803	15	12.867	164804	15	12.367
166318	15	10.417	168579	31	19.300	174921	12	12.500	178012	23	18.966
180657	36	33.450	181709	13	13.083	181754	17	12.067	181761	29	23.283
182711	22	20.900	182712	19	15.200	182713	18	14.100	183531	24	19.350
185854	12	7.184	185860	16	8.866	185864	18	10.066	185866	15	7.917
188716	14	8.650	209362	19	14.050	209862	29	27.134	210192	21	23.550
210455	28	33.117	210593	10	6.166	213528	14	12.250	213529	14	12.283
213795	13	8.600	213798	20	13.000	213799	13	8.684	213853	8	5.733
213855	19	19.833	213856	18	14.383	213857	17	13.783	215669	23	19.566
216382	16	17.500	216383	20	21.483	217938	17	12.067	217947	31	22.500
218178	20	11.433	218179	14	8.100	218183	26	17.917	218184	14	7.800
218185	15	8.283	218186	19	14.283	218187	16	8.933	218188	16	15.367
218189	12	7.367	218915	19	15.417	218997	11	8.467	218999	15	18.500
219075	28	36.050	219079	27	35.083	219539	10	8.033	219551	12	9.183
219553	16	14.017	219554	15	11.433	219555	18	12.750	219557	16	14.350
219558	17	16.400	219907	37	61.967	220152	14	6.834	220748	26	31.367
221353	27	31.517	221354	27	31.450	223156	17	14.133	224159	10	11.883
224161	14	14.733	224163	15	15.550	224164	15	15.416	224547	10	7.384
224548	9	6.933	224549	11	9.316	224550	10	7.400	224551	15	11.900
224552	16	10.900	225194	32	28.233	225196	35	31.983	226385	16	16.900
226386	22	22.200	227282	29	26.483	227283	17	15.283	227284	18	15.917
227285	17	13.167	227286	17	12.483	227462	26	25.800	228544	24	14.917
228545	24	14.833	228546	23	13.983	228795	20	18.816	228796	20	17.500
228797	19	20.634	230834	26	22.650	231373	35	31.950	231374	48	44.150
231375	41	39.550	232744	26	26.117	232745	32	29.133	232747	31	27.516
234050	7	5.800	234064	15	8.350	234756	38	30.450	237492	30	22.533
238225	25	25.266	238226	32	27.917	239091	9	6.966	239092	10	7.784
239093	18	21.900	239094	12	14.850	239095	10	10.017	239097	10	6.983
239098	10	7.050	239126	35	23.750	241426	15	10.133	241427	10	7.917
241452	14	10.767	241950	16	12.433	242099	24	22.300	242109	20	17.600
242112	25	20.667	247152	20	12.767	247153	17	10.000	247155	16	10.067
247181	6	4.750	247186	6	4.984	247740	22	19.117	247741	22	19.150
248387	10	8.933	249254	16	14.250	250629	26	19.000	250630	20	14.333
251272	34	30.734	251701	36	35.967	251703	40	44.150	251704	42	37.600
251706	36	33.717	251707	33	32.433	251708	35	33.700	251709	36	34.066
251710	36	34.083	251855	15	11.417	252492	23	20.650	252494	26	21.017
252496	27	24.667	253959	33	23.100	253960	35	23.600	253961	42	29.567
253962	44	30.750	255577	27	25.200	255578	31	28.333	255580	25	24.300
255655	15	12.634	256393	36	30.400	256394	40	33.650	256395	39	33.934
256979	23	19.000	256999	34	30.833	257009	29	21.684	257019	31	22.800
257055	22	21.417	257110	37	42.833	257135	25	30.717	257274	32	44.300
258795	28	34.500	262865	10	6.950	262866	9	6.050	262867	10	6.833
262868	21	19.900	262869	21	19.834	262870	20	18.866	262871	17	16.183
262872	17	16.133	262873	17	16.234	264968	10	8.850	29263	17	9.267
29293	22	16.083	29773	29	19.050	3204	12	9.500	34070	30	24.367
34271	21	16.617	38239	8	4.866	40226	13	7.700	41136	15	12.650
43176	29	32.250	43694	25	20.000	44263	16	17.667	45935	6	3.366
45936	5	3.250	45937	8	4.216	5824	22	14.734	62439	13	9.417
62440	11	7.017	62441	13	8.500	64287	21	15.383	64288	32	21.583
64289	29	22.034	64472	9	6.717	64473	9	6.916	64474	10	5.800
65019	15	9.250	66694	9	7.234	66695	9	4.400	66696	9	5.484
69812	25	19.917									

2. PHOTO LAB TEST QUERIES USING VISUAL KEYPHRASES

In computing the *recall*, *fallout*, and *precision* statistics for the keyphrase queries, replace each occurrence of "MARIE" in the definitions given earlier with "KEYPHRASE SYSTEM." Caption match results are indicated by the list identified by C.

Formulating the correct keyphrase requires a great deal of domain knowledge about the subject area. Without this knowledge, a naive user would require more specific information and be involved in a more lengthy search.

Query 1.1 *all negs released*

Note. No keyphrase for this; this reverts to a sequential search for the string pattern {R}.

Query 1.2 *exterior view of boxcar*

railroad boxcar

C = []

RECALL = 0/1 = 0; FALLOUT = 0; PRECISION = 1.0

Note. The caption is indexed in the keyphrases as follows:

181761-70 bomb inert roseville boxcar cookoff clpl {r}

Query 1.3 *pictured published in major accomplishments*

graphics publication nwc major accomplishments

C = [181761-70]

RECALL = 1/1 = 1; FALLOUT = 0; PRECISION = 1.0

Query 1.4 *spin canister on a helo*

aircraft XXXX YYYY

where XXXX is the type of helo and YYYY is the type of canister.

Note. Not locatable.

Query 1.5 *picture of a pedestal taken between 1975 and 1976* (involves Reg data)

Note. Not locatable.

Query 1.6 *VX-5 aircraft carrying weapons, in the air*

aircraft XXXX YYYY

where XXXX is the type of aircraft and YYYY is the type of weapon.

Note. Not locatable.

Query 1.7 *DSU-28/B*

fuze dsu-28/b

C = [210192-99]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.8 *training missiles on a Skyhawk*

aircraft a-4* XXXX

where XXXX is the type of missile.

Note. Not locatable.

Query 1.9 *tower at RCC*

facility range rcc tower

C = []

RECALL = 0/1 = 0; FALLOUT = 0; PRECISION = 1.0

Note. The caption is indexed in the keyphrases as follows:

```
213528-29    building 31455 rcc constr af {r}
213528-29    facility rcc constr {r}
213528-29    tower microwave rcc af {r}
```

Query 1.10 *TP 1314 synchro firing at Snort on 2-23-81*

Note. Not locatable.

Query 1.11 *LWIR with a MIG-21 target*

target a/c model mig-21
program lwir
optical ir lwir lights mig-21

Note. Either of the three keyphrases will work.

C = [218997-99]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.12 *J-52 test setup*

test wpns surv j52 fuel ingestion setup

Note. Required additional information from the customer.

C = [219539-42, 234050]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0

Note. "test setup" is *implicitly* known in 234050.

Query 1.13 *personnel soldering*

personnel soldering

```
C = [219551-58]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. In the individual photographs, only 219551, 219554, and 219557 express the relationship that a person is actually engaged in a soldering event.

Query 1.14 *Wild Weasel with two or more weapons*

```
aircraft f-4* wpns uploaded
```

Note. We can also search under specific weapons types.

```
C = [219907-08]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.15 *A-7E dropping bombs*

```
aircraft a-7e XXXX drop
bombs XXXX a-7e drop
firing XXXX a-7e

     where XXXX is the type of bomb.
```

Note. Not locatable.

Query 1.16 *Sidewinder 9C firing against a ground target*

```
firing sidewinder aim-9c

C = []

RECALL = 0/1 = 0; FALLOUT = 0; PRECISION = 1.0
```

Note. The caption is indexed in the keyphrases as follows:

```
221353-54       firing sidearm f-4g tank air/ground {r}
221353-54       missile sidearm firing air/ground {r}
```

Query 1.17 *F/A-18 simulator in hangar 3*

```
airfield af wssa sim lab f/a-18*

C = [226385-86]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.18 *F/A-18 flying over snow capped mountain*

```
aircraft f/a-18* XXXX

   where XXXX is the location
```

Note. Not locatable.

Query 1.19 *Harm missile or Silver Bullet on an aircraft*

```
missile harm XXXX
```

where XXXX is the type of aircraft

Note. Not locatable.

Query 1.20 *GBU-22 on a stand*

```
bomb lllgb gbu-22/b stand
```

C = [234056-71]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.21 *fireball at Weapons Survivability*

```
test wpns surv * fireball
firing wpns surv * fireball
```

Note. Either of the two keyphrases will work.

C = [234050]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.22 *aircraft flying over Armitage airfield*

```
aircraft XXXX armitage
```

where XXXX is the type of aircraft

Note. Not locatable.

Query 1.23 *Sidearm missile published in Aviation Week magazine*

```
graphics publication aviation week sidearm
```

C = [241425-29]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.24 *9R seeker with dome*

```
optical seeker sidewinder aim 9r
electronics sidewinder aim 9r seeker
```

Note. Either of the two keyphrases will work.

C = [247181-88]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.25 *Naces seat ejection*

```
firing naces ejection seat
vehicle sled naces
```

Note. Either of the two keyphrases will work.

C = [251272]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.26 *underside view of aircraft with BQM drones*

target a/c drone bqm*

C = [161044-45]

RECALL = 1/2 = 0.5; FALLOUT = 0; PRECISION = 0

Query 1.27 *Sidewinder 9M missiles being launched*

firing sidewinder aim-9m

C = []

firing sidewinder aim 9m

C = [216381-82, 252494, 252496-97]

RECALL = 3/4 = 0.75; FALLOUT = 0; PRECISION = 1.0

Query 1.28 *BTV on an A-7 aircraft*

missile walleye i btv ta-7c skyray fiber optics pod
pod skyray fiber optics ta-7c walleye btv

Note. Either of the two keyphrases will work.

C = [253959-63]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.29 *front view of an aircraft on the ground or in the air*

Note. Not locatable.

Query 1.30 *Skyray pod closeup view*

pod skyray fiber optics

C = [253959-63, 256979-99]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.31 *night attack aircraft*

program night attack

C = [257009-19, 257110-19, 257135]

RECALL = 3/3 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.32 *Assault Breaker tests*

vehicle sled assault breaker

C = [209362, 217942-49]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.33 *VX-5 and NWC aircraft flying together*

Note. Not locatable.

Query 1.34 *night vision goggles being worn*

optical goggles night vision

C = [257135, 258795]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.35 *firing of a Hipeg MK 4 gun*

firing hipeg mk 4 gun

C = [29293]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.36 *camera and operator in the Machine Shop*

Note. Not locatable.

Query 1.37 *Sidewinder missile comparison*

missile sidewinder

C = [161082-84, 182711-13, 216381-82, 242099, 251700-10, 252492, 252494, 252496-97, 255577-80, 256393-95, 257055, 258795, 262865-73, 264966-68, 3204, 5824, 29293-95, 29773]

RECALL = 1/1 = 1.0; FALLOUT = 17/18 = 0.94; PRECISION = 0.06

Query 1.38 *CLAM B bombs*

bomb fireye clam b

C = [34271]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.39 *S-2A aircraft with pod*

aircraft s-2a

C = [40226]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.40 *open parachutes from a helo*

Note. Not locatable.

Query 1.41 *side view of an aircraft with Fat Albert*

Note. Not locatable as we would need the type of aircraft or do a search under Walleye.

Query 1.42 *Bullpup target*

Note. Not locatable as we would need more information on the type of weapon, etc.

Query 1.43 *Tomcat launching Phoenix missile*

firing phoenix f-14*

C = [64287-89]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.44 *solar intensity studies*

environmental solar intensity

C = [64472-74]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.45 *Hot Line at NAF*

airfield naf hot line

C = [66694-96]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.46 *photovoltaic panels*

environmental solar

C = [64472-74, 69812]

RECALL = 1/1 = 1.0; FALLOUT = 1/2 = 0.5; PRECISION = 0.5

3. PHOTO LAB TEST QUERIES USING MARIE

The keyphrase queries were based on captions that describe a set of photographs (supercaptions) for a particular photo sequence. In this section, each photo was described by its own caption to allow retrieval of an individual photograph instead of a set of photographs.

Included with the retrieval effectiveness measures are the real-time parsing times and a list of real-time match times for MARIE on a network of Sparcstation workstations. Six tests were conducted. The first number (1*) reports the elapsed time where one Sparcstation 2 was running one MarieSearch, one MarieNLP, and one MarieFine process. The last five numbers show the results when MarieFine was executing on from one to five different Sparcstations respectively. A single Sparcstation 1+ was used as a file server to hold the database and represents a potential bottleneck. In the list of captions retrieved, C, the first number indicates the caption score and the second shows the caption identifier.

Query 1.1 *all negs released*

This information (i.e., {R}) is currently embedded in the *keyphrase* field of the *KEYPHRASES* relation. A more appropriate means to handle this information and query is to add an additional field (e.g., *released*) to the *VISUAL* relation. The images can then be found via the SQL command:

```
select id
from visual
where released = "R"
```

Query 1.2 *exterior view of boxcar*

```
%PARSE_STATS = 2.184 sec.
%MAX_SCORE = 4.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [5.532,5.264,5.350,5.422,5.118,5.567] sec.

C = [4.0-181761]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.3 *pictured published in major accomplishments*

This information (i.e., publication information) is currently embedded in the *caption* field of the *VISUAL* relation. A more appropriate means to handle this information and query is to add the following additional relation:

```
create table publications (  designator varchar(6) not null,
                             id varchar(9) not null,
                             pub_name varchar(64) not null,
                             pub_issue varchar(8),
                             pub_date date );
```

The images can then be retrieved via the SQL command:

```
select id
from publications
```

Query 1.4 *spin canister on a helo*

```
%PARSE_STATS = 2.433 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [9.719,6.698,4.864,4.915,4.003,4.018] sec.

C = [3.0-185860, 3.0-185866, 3.0-185864, 3.0-185854]

RECALL = 4/4 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.5 *picture of a pedestal taken between 1975 and 1976* (involves Reg data)

```
%PARSE_STATS = 2.317 sec.
%MAX_SCORE = 11.0
%KEYWORD_SCORE = 4.0
%SEARCH_STATS = [4.743,4.508,4.634,4.802,4.461,4.393] sec.

C = [3.0-188716]
```

```
RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. This query requires examining both the parsed caption and the registration data (specifically the *date_orig* attribute) in the *VISUAL* relation. The SQL command inferred by the query structure is:

```
select id
from visual
where 1975 <= date_orig and date_orig <= 1976
```

These identifiers would then be intersected with the identifiers found in the coarse-grain search (keyword only) to derive those identifiers that meet the coarse-grain threshold and are suitable for fine-grain matching. The date range comparison has not been implemented which is why we have such a low score. The above identifiers are derived from the query: *picture of a pedestal.*

<u>Query 1.6</u> *VX-5 aircraft carrying weapons, in the air*

```
Rephrase as: air to air view of VX-5 aircraft carrying weapons
         or: VX-5 aircraft (air-to-air view) carrying weapons

%PARSE_STATS = 5.766 sec.
%MAX_SCORE = 12.0
%KEYWORD_SCORE = 4.0
%SEARCH_STATS = [76.578,50.746,39.412,39.724,40.199,39.593] sec.

C = [12.0-209862, 6.0-257274]

RECALL = 1/2 = 0.5; FALLOUT = 0; PRECISION = 1.0
```

Note. The parser cannot handle the query structure with the comma. By restating the query in one of the two forms below, a better answer will be produced. A query rephrase of "VX-5 aircraft in the air carrying weapons" leads to an incorrect interpretation because of the way participle structures are handled in the functional parse rules.

<u>Query 1.7</u> *DSU-28/B*

```
%PARSE_STATS = 0.867 sec.
%MAX_SCORE = 1.0
%KEYWORD_SCORE = 1.0
%SEARCH_STATS = [0.668,0.714,0.782,0.666,0.545,0.385] sec.

C = [1.0-210192]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

<u>Query 1.8</u> *training missiles on a Skyhawk*

```
%PARSE_STATS = 4.500 sec.
%MAX_SCORE = 6.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [12.419,11.252,11.235,11.051,11.244,10.683] sec.

C = [5.5-210455]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

<u>Query 1.9</u> *tower at RCC*

```
%PARSE_STATS = 1.783 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [3.642,3.166,2.514,2.403,2.106,2.339] sec.

C = [3.0-213529, 3.0-213528]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. This query requires examining both the parsed caption and the registration data (specifically the *location* attribute) in the *VISUAL* relation. The SQL command inferred by the query structure is:

```
select id
from visual
where location = "Range Control Center"
```

(RCC is being mapped to Range Control Center in the lexicon)

These identifiers are then intersected with the identifiers found in the coarse-grain search (keyword only) to derive those identifiers that meet the coarse-grain threshold and are suitable for fine-grain matching.

Query 1.10 *TP 1314 synchro firing at Snort on 2-23-81*

Rephrase as: *TP 1314. synchro firing at Snort on Feb 23, 1981*

```
%PARSE_STATS = 6.400 sec.
%MAX_SCORE = 9.0
%KEYWORD_SCORE = 4.0
%SEARCH_STATS = [5.426,4.255,4.186,4.187,4.344,4.183] sec.

C =  [9.0-215669]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. The query can also be shortened to: *TP 1314.*

```
%PARSE_STATS = 1.300 sec.
%MAX_SCORE = 2.0
%KEYWORD_SCORE = 1.0
%SEARCH_STATS = [59.432,30.628,16.323,11.095,8.468,7.885] sec.

C = [2.0-215669, 1.0-255580, 1.0-255578, 1.0-255577, 1.0-252496, 1.0-
252494, 1.0-252492,  1.0-251710, 1.0-251709, 1.0-251708, 1.0-251707, 1.0-
251706, 1.0-251704,  1.0-251703,  1.0-251701, 1.0-251272, 1.0-250630, 1.0-
250629, 1.0-237492, 1.0-221354, 1.0-221353, 1.0-219539, 1.0-218915, 1.0-
217947, 1.0-209362, 1.0-180657,  1.0-3204]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. This query requires examining both the parsed caption and the registration data (specifically the *location* and *date_orig* attributes) in the *VISUAL* relation. The SQL command inferred by the query statement is:

```
select id
from visual
where location = "Snort"
and date_orig = 23-feb-1981
```

These identifiers are then intersected with the identifiers found in the coarse-grain search (keyword only) to derive those identifiers that meet the coarse-grain threshold and are suitable for fine-grain matching.

Query 1.11 *LWIR with a MIG-21 target*

```
%PARSE_STATS = 6.550 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [3.816,3.588,3.646,3.753, 3.684,3.622] sec.

C = [4.0-218999]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.12 *J-52 test setup*

```
%PARSE_STATS = 3.850 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [3.834,3.476,3.412,3.786,3.766,3.606] sec.

C = [4.0-219539]

RECALL = 1/2 = 0.5; FALLOUT = 0; PRECISION = 1.0
```

Note. 234050 should also be retrieved because "test setup" is implicitly known to have occurred in the caption event.

Query 1.13 *personnel soldering*

```
%PARSE_STATS = 2.117 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 1.0
%SEARCH_STATS = [134.505,70.057,37.694,27.206,21.951,18.142] sec.

C = [3.0-219554, 3.0-219551, 3.0-219557, 2.0-219555, 2.0-219553, 2.0-
219558, 1.0-251855, 1.0-227283, 1.0-227282, 1.0-218189, 1.0-218187,
1.0E+00-218186, 1.0-218185, 1.0-218184, 1.0-218183, 1.0-218179, 1.0-213855,
1.0-181754, 1.0-168579, 1.0-257135, 1.0-258795, 1.0-218188, 1.0-257055,
1.0-69812, 1.0-226385, 1.0-227462, 1.0-226386, 1.0E+00-239093, 1.0-239094,
1.0-239095, 1.0-220748, 1.0-228544, 1.0-228546, 1.0-228545, 1.0-185864,
1.0-257110, 1.0-10880, 1.0-62440, 1.0-85489, 1.0-124]

RECALL = 3/3 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.14 *Wild Weasel with two or more weapons*

Rephrase as: *Wild Weasel with multiple weapons*

```
%PARSE_STATS = 4.416 sec.
%MAX_SCORE = 4.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [42.331,28.192,20.740,18.234,17.741,17.871] sec.

C = [4.0-219907, 2.0-221353, 2.0-221354, 2.0-228795, 2.0-228796, 2.0-
228797, 2.0-249254]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. The parser produced an erroneous parse tree for the original query because of the "or" structure.

Query 1.15 *A-7E dropping bombs*

```
%PARSE_STATS = 2.933 sec.
%MAX_SCORE = 6.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [32.170,19.014,12.277,10.031,9.969,9.764] sec.

C = [6.0-220748, 3.0-209862, 3.0-247155, 3.0-247152, 3.0-247153, 2.0-168579]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.16 *Sidewinder 9C firing against a ground target*

```
%PARSE_STATS = 6.683 sec.
%MAX_SCORE = 7.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [15.698,10.083,6.866,6.649,6.899,6.883] sec.

C = [7.0-221354, 7.0-221353]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.17 *F/A-18 simulator in hangar 3*

```
%PARSE_STATS = 2.900 sec.
%MAX_SCORE = 4.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [6.535,4.671,3.078,3.199,3.149,3.046] sec.

C = [3.5-226386, 3.5-226385]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.18 *F/A-18 flying over snow capped mountain*

```
%PARSE_STATS = 4.916 sec.
%MAX_SCORE = 8.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [9.440,6.604,6.894,6.559,6.692,6.353] sec.

C = [3.0-227462]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.19 *Harm missile or Silver Bullet on an aircraft*

Rephrase as two queries:
a. *Harm missile on an aircraft*
b. *Silver Bullet on an aircraft*

```
a. %PARSE_STATS = 2.933 sec.
   %MAX_SCORE = 3.0
   %KEYWORD_SCORE = 2.0
   %SEARCH_STATS = [77.560,45.947,29.851,24.878,21.762,20.926] sec.

   C = [3.0-231373, 3.0-231374, 3.0-231375, 3.0-232744, 3.0-232745, 3.0-
   232747, 3.0-258795, 3.0-219907, 3.0-257274, 3.0-230834, 3.0-242109, 3.0-
```

```
242112]

RECALL = 12/12 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

b.
```
%PARSE_STATS = 2.317
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [27.055,20.548,17.366,15.453,15.162,15.390] sec.

C = [3.0-231374, 3.0-231375, 3.0-231373]

RECALL = 3/3 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. The parser produced an erroneous parse tree for the original query because of the "or" structure.

Query 1.20 *GBU-22 on a stand*

```
%PARSE_STATS = 2.683 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [2.776,2.872,2.868,2.753,2.816,2.991] sec.

C = [3.0-234064]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.21 *fireball at Weapons Survivability*

```
%PARSE_STATS = 1.833 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [1.962,1.899,1.896,1.823,1.899,2.189] sec.

C = [2.0-234050]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. This query requires examining both the parsed caption and the registration data (specifically the *location* attribute) in the VISUAL relation. The SQL command inferred by the query structure is:

```
select id
from visual
where location = "Weapons Survivability Laboratory"
```

(Weapons Survivability is being mapped to Weapons Survivability Laboratory in the lexicon)

These identifiers are then intersected with the identifiers found in the coarse-grain search (keyword only) to derive those identifiers that meet the coarse-grain threshold and are suitable for fine-grain matching.

Query 1.22 *aircraft flying over Armitage airfield*

```
%PARSE_STATS = 2.917 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [23.640,18.669,15.791,15.102,15.783,15.763] sec.
```

C = [2.0-234756, 2.0-239126, 2.0-66694]

RECALL = 2/2 = 1.0; FALLOUT = 1/3 = 0.333; PRECISION = 0.667

Query 1.23 *Sidearm missile published in Aviation Week magazine*

%PARSE_STATS = 1.484 sec.
%MAX_SCORE = 1.0
%KEYWORD_SCORE = 1.0
%SEARCH_STATS = [1.591,1.744,1.673,1.270,1.446,1.419] sec.

C = [1.0-241950, 1.0-241427, 1.0-241426, 1.0-219079, 1.0-221353, 1.0-221354, 1.0-219075]

RECALL = 2/2 = 1.0; FALLOUT = 5/7 = 0.71; PRECISION = 0.29

Note. This information (i.e., publication information) is currently embedded in the *caption* field of the *VISUAL* relation and in the *KEYPHRASES* relation. Query 1.3 described a more appropriate means to handle this information and query. If the publications table exists, then we could issue the following query

```
select id
from publications
where pub_name = "Aviation Week"
```

These identifiers would then be intersected with the identifiers found in the coarse-grain search (keyword only) to derive those identifiers that meet the coarse-grain threshold and are suitable for fine-grain matching. The above identifiers are derived from the query: *Sidearm missile.*

Query 1.24 *9R seeker with dome*

Rephrase as: *AIM 9R seeker with dome*

%PARSE_STATS = 3.783 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [5.751,4.326,2.957,3.025,3.144,3.043] sec.

C = [5.0-247186, 4.5-247181]

RECALL = 1/1 = 1.0; FALLOUT = 1/2 = 0.5; PRECISION = 0.5

Query 1.25 *Naces seat ejection*

%PARSE_STATS = 3.200 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [4.121,3.361,3.284,3.358,3.306,3.298] sec.

C = [5.0-251272]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.26 *underside view of aircraft with BQM drones*

%PARSE_STATS = 5.633 sec.
%MAX_SCORE = 8.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [14.849,15.416,14.893,15.395,15.212,15.121] sec.

```
C = []

RECALL = 0/1 = 0; FALLOUT = 0; PRECISION = 1.0
```

Rephrasing the query as: *view under aircraft with BQM drones*

```
%PARSE_STATS = 4.683 sec.
%MAX_SCORE = 6.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [21.099,18.734,16.970,16.901,16.853,16.816] sec.

C = [4.0-252492, 3.0-161045]

RECALL = 1/2 = 0.5; FALLOUT = 0; PRECISION = 1.0
```

Note. In 161045, the fact that the drone is on an aircraft and we have a view of the drone is implicitly understood to mean that it is also under the aircraft.

Query 1.27 *Sidewinder 9M missiles being launched*

```
%PARSE_STATS = 3.183 sec.
%MAX_SCORE = 4.0
%KEYWORD_SCORE = 1.0
%SEARCH_STATS = [24.655,14.125,8.056,6.408,5.268,4.729] sec.

C = [3.5-252492, 3.0-252496, 3.0-252494, 3.0-216382, 1.0-255580, 1.0-
255578,  1.0-255577]

RECALL = 4/4 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.28 *BTV on an A-7 aircraft*

```
%PARSE_STATS = 4.084 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [24.493,13.942,7.991,7.473,5.215,5.566] sec.

C = [3.0-253959, 2.0-253961, 2.0-253962, 2.0-253960]

RECALL = 1/4 = 0.25; FALLOUT = 0; PRECISION = 1.0
```

Query 1.29 *front view of an aircraft on the ground or in the air*

```
%PARSE_STATS = 2.100 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
2%SEARCH_STATS = [42.046,27.996,20.445,18.275,17.173,16.031] sec.

C = [3.0-238225, 3.0-256393, 3.0-257110, 3.0-262871, 3.0-262872, 3.0-
213856,  3.0-256979, 2.0-256394, 2.0-227286, 2.0-262869]

RECALL = 7/7 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. The parser cannot handle "or" correctly. Query can be rephrased as was shown in Query 1.19 or simplified to *front view of an aircraft* (whose results are shown above) which should produce the same results as the original query.

Query 1.30 *Skyray pod closeup view*

```
%PARSE_STATS = 3.450 sec.
```

%MAX_SCORE = 4.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [30.061,17.782,11.293,8.534,8.621,7.248] sec.

C = [4.0-256999, 3.0-256979, 2.0-253960, 2.0-253961, 2.0-253962, 2.0-253959]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Note. In 256979 and 253960, the fact that we have a "closeup view" could be inferred because of "3/4 stbd front view" (in 256979) and "view below aircraft" (in 253960).

Query 1.31 *night attack aircraft*

%PARSE_STATS = 1.850 sec.
%MAX_SCORE = 2.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [300.965,157.513,84.884,61.622,49.122,42.728] sec.

C = [2.0-257110, 2.0-257019, 2.0-257009, 2.0-257135, 1.0-218189, 1.0-218185, 1.0-218179, 1.0-66694, 1.0-210192, 1.0-250630, 1.0-250629, 1.0-239126, 1.0-238226, 1.0-238225, 1.0-43694, 1.0-181754, 1.0-110169, 1.0-242109, 1.0-257274, 1.0-242112, 1.0-209862, 1.0-215669, 1.0-218915, 1.0-213855, 1.0-253962, 1.0-253961, 1.0-253960, 1.0-253959, 1.0-213857, 1.0-213856, 1.0-164803, 1.0-247155, 1.0-247153, 1.0-247152, 1.0-220748, 1.0-181709, 1.0-168579, 1.0-164804, 1.0-163030, 1.0-219079, 1.0-219075, 1.0-180657, 1.0-225194, 1.0-241452, 1.0-237492, 1.0-225196, 1.0-900304, 1.0-64287, 1.0-216383, 1.0-64289, 1.0-64288, 1.0-217938, 1.0-219907, 1.0-221354, 1.0-221353, 1.0-227282, 1.0-226385, 1.0-258795, 1.0-25
7055, 1.0-256999, 1.0-256979, 1.0-256395, 1.0-256394, 1.0-256393, 1.0-255580, 1.0-255578, 1.0-255577, 1.0-252496, 1.0-252494, 1.0-252492, 1.0-247741, 1.0-247740, 1.0-232747, 1.0-232745, 1.0-232744, 1.0-231375, 1.0-231374, 1.0-231373, 1.0-230834, 1.0-227462, 1.0-227286, 1.0-227285, 1.0-227284, 1.0-227283, 1.0-226386, 1.0-251701, 1.0-262873, 1.0-262872, 1.0-262871, 1.0-262870, 1.0-262869, 1.0-262868, 1.0-251710, 1.0-251709, 1.0-251708, 1.0-251707, 1.0-251706, 1.0-251704, 1.0-251703, 1.0-228795, 1.0-228797, 1.0-228796, 1.0-234756, 1.0-182711, 1.0-264968, 1.0-216382, 1.0-182713, 1.0-182712, 1.0-85486, 1.0-85489, 1.0-85488, 1.0-85487, 1.0-251272, 1.0-3204, 1.0-5824, 1.0-10862, 1.0-218183, 1.0-40226, 1.0-218178, 1.0-34070, 1.0-29293, 1.0-43176, 1.0-34271, 1.0-210455, 1.0-161045, 1.0-249254, 1.0-218999, 1.0-241950, 1.0-242099, 1.0-161082, 1.0-251855, 1.0-41136, 1.0-185854, 1.0-213799, 1.0-213798, 1.0-213795, 1.0-185866, 1.0-185864, 1.0-185860, 1.0-161044, 1.0-124, 1.0-219539]

RECALL = 4/4 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.32 *Assault Breaker tests*

%PARSE_STATS = 2.133 sec.
%MAX_SCORE = 4.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [6.158,4.769,3.640,3.653,3.574,3.685] sec.

C = [3.0-217947, 3.0-209362]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.33 *VX-5 and NWC aircraft flying together*

%PARSE_STATS = 5.417 sec.

%MAX_SCORE = 8.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [30.231,21.552,21.689,21.847,21.669,22.908] sec.

C = [4.0-257274]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.34 *night vision goggles being worn*

%PARSE_STATS = 2.667 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [11.426,6.845,4.227,4.425,4.381,4.687] sec.

C = [5.0-258795, 5.0-257135]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.35 *firing of a Hipeg MK 4 gun*

%PARSE_STATS = 3.550 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [2.805,2.494,2.458,2.745,2.550,2.772] sec.

C = [3.0-29293]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.36 *camera and operator in the Machine Shop*

Note. The parse tree produced for this query is partially erroneous. See Chapter IV for the problem of handling conjunctive nouns.

%PARSE_STATS = 3.100 sec.
%MAX_SCORE = 4.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [3.013,2.928,2.687,2.879,2.861,2.901] sec.

C = [3.0-29263]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.37 *Sidewinder missile comparison*

%PARSE_STATS = 3.367 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [6.881,5.346,5.205,5.209,4.933,5.527] sec.

C = [3.0-29773]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.38 *CLAM B bombs*

%PARSE_STATS = 1.966 sec.
%MAX_SCORE = 2.0
%KEYWORD_SCORE = 2.0

%SEARCH_STATS = [2.237,2.090,1.946,1.954,2.025,2.176] sec.

C = [2.0-34271]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.39 *S-2A aircraft with pod*

%PARSE_STATS = 3.600 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [3.008,2.943,2.987,2.892,2.914,2.927] sec.

C = [3.0-40226]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.40 *open parachutes from a helo*

%PARSE_STATS = 2.950 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [3.983,3.773,3.653,3.682,3.640,3.922] sec.

C = [3.0-41136]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.41 *side view of an aircraft with Fat Albert*

%PARSE_STATS = 5.383 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 3.0
%SEARCH_STATS = [38.958,27.419,20.495,19.247,17.276,17.428] sec.

C = [5.0-253959, 5.0-43176, 4.0-253961, 4.0-253962]

RECALL = 2/4 = 0.5; FALLOUT = 0; PRECISION = 1.0

Note. "view below aircraft" in 253960 might also be inferred as being a side view because three of the four pictures in the photo sequence mention side view.

Query 1.42 *Bullpup target*

%PARSE_STATS = 2.733 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [2.621,2.456,2.402,2.429,2.394,2.473] sec.

C = [3.0-44263]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 1.43 *Tomcat launching Phoenix missile*

%PARSE_STATS = 3.550 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [20.823,12.252,7.140,6.671,5.031,5.046] sec.

```
C = [5.0-64288, 5.0-64289, 5.0-64287, 2.0-216383]

RECALL = 3/3 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.44 *solar intensity studies*

```
%PARSE_STATS = 2.116 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [4.732,3.292,3.128,2.053,1.953,2.091] sec.

C = [5.0-64474, 5.0-64473, 5.0-64472]

RECALL = 3/3 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 1.45 *Hot Line at NAF*

```
%PARSE_STATS = 1.783 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [2.046,2.112,1.999,2.126,2.161,2.084] sec.

C = [3.0-66695]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. This query requires examining both the parsed caption and the registration data (specifically the *location* attribute) in the *VISUAL* relation. The SQL command inferred by the query structure is:

```
select id
from visual
where location = "Armitage Field"
```

(NAF is being mapped to Armitage Field in the lexicon)

These identifiers are then intersected with the identifiers found in the coarse-grain search (keyword only) to derive those identifiers that meet the coarse-grain threshold and are suitable for fine-grain matching.

Query 1.46 *photovoltaic panels*

```
%PARSE_STATS = 1.700 sec.
%MAX_SCORE = 4.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = [3.542,2.836,2.758,2.867,2.810,3.059] sec.

C = [4.0-69812]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

4. OUR TEST QUERIES USING MARIE

The search stat numbers report the elapsed time where one Sparcstation 2 is running one MarieSearch and one MarieNLP process, and a separate Sparcstation 2 was running a MarieFine process. A single Sparcstation 1+ was used as a file server to hold the database.

Query 2.1 *Sidewinder.*

```
%PARSE_STATS = 1.100 sec.
%MAX_SCORE = 1.0
%KEYWORD_SCORE = 1.0
%SEARCH_STATS = 1.300 sec.

C = [1.0-258795, 1.0-5824, 1.0-251701, 1.0-251710, 1.0-251709, 1.0-251708,
1.0-251707, 1.0-251706, 1.0-251704, 1.0-251703, 1.0-29773, 1.0-161082, 1.0-
256395, 1.0-256394, 1.0-256393, 1.0-242099, 1.0-182713, 1.0-182712, 1.0-
182711, 1.0-216382, 1.0-255580, 1.0-255578, 1.0-255577, 1.0-252496, 1.0-
252494, 1.0-252492, 1.0-247181, 1.0-264968, 1.0-262873, 1.0-262872, 1.0-
262871, 1.0-262870, 1.0-262869, 1.0-262868, 1.0-262867, 1.0-262866, 1.0-
262865, 1.0-257055, 1.0-247186, 1.0-221353, 1.0-221354, 1.0-3204]

RECALL = 42/42 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. This query finds all of the captions that contain the term "Sidewinder" or one of its subclasses (e.g., "AIM-9B," "AIM-9R " etc.).

Query 2.2 *Sidewinder mounted on a st*

```
%PARSE_STATS = 3.734 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 11.674 sec.

C = [3.0-262865, 3.0-262867, 3.0-262866, 3.0-29773]

RECALL = 4/4 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. The maximum possible match score for each of the four captions is three.

Query 2.3 *Sidewinder on F/A-18.*

```
%PARSE_STATS = 2.983 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 64.877 sec.

C = [3.0-252492, 3.0-252494, 3.0-252496, 3.0-255577, 3.0-255578, 3.0-
255580,  3.0-256393, 3.0-256394, 3.0-256395, 3.0-257055, 3.0-258795, 3.0-
251701, 3.0-251703, 3.0-251704, 3.0-251706, 3.0-251707, 3.0-251708, 3.0-
251709, 3.0-251710, 3.0-262868, 3.0-262869, 3.0-262870, 3.0-262871, 3.0-
262872, 3.0-262873, 3.0-264968]

RECALL = 26/26 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. This type of query required introduction of the transformation rules.

Query 2.4 *Sidearm hitting a target.*

```
%PARSE_STATS = 2.883 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 8.578 sec.

C = [5.0-241427, 2.0-221353, 2.0-221354, 2.0-241426]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 2.5 *missiles fired from an aircraft.*

```
%PARSE_STATS = 2.850 sec.
%MAX_SCORE = 6.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 209.610 sec.

C = [5.5-252492, 5.0-64287, 5.0-64288, 5.0-64289, 5.0-221353, 5.0-221354,
5.0-242109, 5.0-242112, 5.0-161082, 5.0-242099, 5.0-216382, 5.0-252494,
5.0-252496, 5.0-257055, 5.0-264968, 5.0-231373, 5.0-231374, 5.0-231375,
5.0-241950, 4.5-10862, 4.0-182711, 4.0-182712, 4.0-182713, 3.0-256394, 3.0-
256395, 3.0-110169, 3.0-258795, 3.0-3204, 3.0-219907, 2.5-251701, 2.5-
251703, 2.5-251704, 2.5-251706, 2.5-251707, 2.5-251708, 2.5-251709, 2.5-
251710, 2.5-210455, 2.0-257274, 2.0-210192, 2.0-228795, 2.0-228796, 2.0-
228797, 2.0-34070, 2.0-238225, 2.0-238226, 2.0-256393, 2.0-255577, 2.0-
255578, 2.0-255580, 2.0-262868, 2.0-262869, 2.0-262870, 2.0-262871, 2.0-
262872, 2.0-262873, 2.0-230834, 2.0-232744, 2.0-232745, 2.0-232747, 2.0-
164803, 2.0-164804, 2.0-247740, 2.0-247741, 2.0-219075, 2.0-219079, 2.0-
5824, 2.0-250629, 2.0-250630, 2.0-249254, 2.0-216383]

RECALL = 19/20 = 0.95; FALLOUT = 0; PRECISION = 1.0
```

Note. 10862 is also a valid answer. However, an aircraft *agent* is not stated in the caption, nor inferred at present.

Query 2.6 *missile hitting drone*

```
%PARSE_STATS = 1.983 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 33.757 sec.

C = [5.0-64288, 5.0-228795, 5.0-64289, 5.0-228796, 5.0-228797, 5.0-264968,
5.0-64287, 4.0-182712, 4.0-216382, 4.0-257055, 2.0-182711, 2.0-182713, 2.0-
252492, 2.0-252494, 2.0-252496]

RECALL = 7/10 = 0.7; FALLOUT = 0; PRECISION = 1.0
```

Query 2.7 *second test*

```
%PARSE_STATS = 1.117 sec.
%MAX_SCORE = 2.0
%KEYWORD_SCORE = 1.0
%SEARCH_STATS = 18.538 sec.

C = [1.0-248387, 1.0-239126, 1.0-218915, 1.0-217947, 1.0-215669, 1.0-
238225, 1.0-238226, 1.0-178012, 1.0-181761, 1.0-219539, 1.0-213795, 1.0-
213799, 1.0-213798, 1.0-210192, 1.0-188716, 1.0-255577, 1.0-255580, 1.0-
255578, 1.0-209362, 1.0-180657]

RECALL = 0/0 = 0; FALLOUT = 20/20 = 1.0; PRECISION = 0
```

Note. A score of 1 is just a keyword score; notice that the best score should be one with 2 matches. A current issue is indicating to the user that we have no best answer. However, we can surmise that a score greater than the keyword score should be good.

Query 2.8 *first test*

```
%PARSE_STATS = 1.116 sec.
```

```
%MAX_SCORE = 2.0
%KEYWORD_SCORE = 1.0
%SEARCH_STATS = 18.219 sec.

C = [2.0-248387, 1.0-239126, 1.0-218915, 1.0-217947, 1.0-215669, 1.0-
238225, 1.0-238226, 1.0-178012, 1.0-181761, 1.0-219539, 1.0-213795, 1.0-
213799, 1.0-213798,  1.0-210192, 1.0-188716, 1.0-255577, 1.0-255580, 1.0-
255578, 1.0-209362,  1.0-180657]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Query 2.9 *missile 100 feet out*

```
%PARSE_STATS = 2.500 sec.
%MAX_SCORE = 4.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 6.869 sec.

C = [2.0-182711]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

QUERY records are:
        attribute(noun(query-1-2),inst(noun(query-1-1))).
        inst(noun(query-1-2),feet).
        attribute(noun(query-1-1),100).
        inst(noun(query-1-1),missile).

CAPTION records (subset) are:
        inst('noun(182711-1-a36)','AIM-9L').
        quantity('noun(182711-2-2)',plural('100')).
        inst('noun(182711-2-2)',feet).
        attribute('be(182711-2-1)',inst('noun(182711-2-2)')).
        theme('be(182711-2-1)',obj('noun(182711-1-a36)')).
        attribute('be(182711-2-1)',out).
        activity('be(182711-2-1)',be).
```

Note.

a. A problem shows up in the query statement where "missile 100" is preferred over "100 feet" because of similar situations like "BU# 123456," "nose 110," "tail 32," "USAF# 11111," etc.
b. The score of 2 reflects the fact that "feet" matched + "missile" and "AIM-9L" matched.
c. Both of these structures are not handled well and the present parse rules and weights give erroneous interpretations that are difficult to adjust properly.

Query 2.10 *missile approaching drone*

```
%PARSE_STATS = 2.067 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 33.231 sec.

C = [2.0-64288, 2.0-228795, 2.0-64289, 2.0-228796, 2.0-228797, 2.0-182711,
2.0-182713, 2.0-182712, 2.0-216382, 2.0-252492, 2.0-252494, 2.0-252496,
2.0-257055,  2.0-264968, 2.0-64287]

RECALL = 4/4 = 1.0; FALLOUT = 11/15 = 0.73; PRECISION = 0.27
```

Note. Score of 2 for all is just keyword score; notice that the best score should be one with 5 matches.

Query 2.11 *F-4 hit by missile*

%PARSE_STATS = 3.200 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 24.850 sec.

C = [5.0-228795, 5.0-228796, 5.0-228797, 2.0-221353, 2.0-221354, 2.0-249254, 2.0-219907]

RECALL = 3/3 = 1.0; FALLOUT = 0; PRECISION = 1.0

Query 2.12 *missile underneath aircraft*

%PARSE_STATS = 1.667 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 112.372 sec.

C = [3.0-110169, 3.0-64288, 2.0-238225, 2.0-238226, 2.0-250629, 2.0-250630, 2.0-242109, 2.0-242112, 2.0-257274, 2.0-164803, 2.0-164804, 2.0-219075, 2.0-219079, 2.0-64287, 2.0-64289, 2.0-216383, 2.0-219907, 2.0-221353, 2.0-221354, 2.0-230834, 2.0-247740, 2.0-252492, 2.0-256393, 2.0-252494, 2.0-256394, 2.0-256395, 2.0-252496, 2.0-255577, 2.0-255578, 2.0-247741, 2.0-255580, 2.0-257055, 2.0-231373, 2.0-231374, 2.0-231375, 2.0-232744, 2.0-232745, 2.0-232747, 2.0-258795, 2.0-251701, 2.0-251703, 2.0-251704, 2.0-262868, 2.0-251706, 2.0-262869, 2.0-262870, 2.0-262871, 2.0-262872, 2.0-262873, 2.0-251707, 2.0-251708, 2.0-251709, 2.0-251710, 2.0-228795, 2.0-228796, 2.0-228797, 2.0-182711, 2.0-182712, 2.0-182713, 2.0-216382, 2.0-264968, 2.0-34070, 2.0-210455, 2.0-3204, 2.0-5824, 2.0-10862, 2.0-249254, 2.0-241950, 2.0-242099, 2.0-161082, 2.0-210192]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0

Note. If we make an assumption that "on," "at," and "under" all express similar type relations with respect to location, then we get the following results.

C = [3.0-257274, 3.0-64287, 3.0-64288, 3.0-64289, 3.0-228795, 3.0-228796, 3.0-228797, 3.0-210192, 3.0-10862, 3.0-221353, 3.0-221354, 3.0-242109, 3.0-242112, 3.0-256393, 3.0-256394, 3.0-256395, 3.0-182711, 3.0-182712, 3.0-182713, 3.0-216382, 3.0-252492, 3.0-252494, 3.0-252496, 3.0-257055, 3.0-262868, 3.0-262869, 3.0-262870, 3.0-262871, 3.0-262872, 3.0-262873, 3.0-264968, 3.0-110169, 3.0-231373, 3.0-231374, 3.0-231375, 3.0-258795, 3.0-3204, 3.0-250629, 3.0-250630, 3.0-249254, 2.5-251701, 2.5-251703, 2.5-251704, 2.5-251706, 2.5-251707, 2.5-251708, 2.5-251709, 2.5-251710, 2.5-34070, 2.5-238225, 2.5-238226, 2.5-210455, 2.5-255577, 2.5-255578, 2.5-255580, 2.5-230834, 2.5-232744, 2.5-232745, 2.5-232747, 2.5-164803, 2.5-247740, 2.5-247741, 2.5-164804, 2.5-219075, 2.5-219079, 2.5-5824, 2.5-216383, 2.5-219907, 2.0-161082, 2.0-242099, 2.0-241950]

Query 2.13 *side view of an A-6*

%PARSE_STATS = 2.700 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 8.656 sec.

C = [3.0-257274, 3.0-181754, 2.0-242109, 2.0-110169]

RECALL = 2/2 = 1.0; FALLOUT = 0; PRECISION = 1.0

<u>Query 2.14</u> *missiles in tests*

%PARSE_STATS = 2.117 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 15.984 sec.

C = [3.0-238226, 3.0-255577, 3.0-255578, 3.0-255580, 3.0-209362, 3.0-238225, 2.0-210192]

RECALL = 6/7 = 0.857; FALLOUT = 0; PRECISION = 1.0

<u>Query 2.15</u> *Sidewinder hitting a target*

%PARSE_STATS = 3.000 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 10.456 sec.

C = [4.0-216382, 2.0-221353, 2.0-221354, 2.0-161082]

RECALL = 1/1 = 1.0; FALLOUT = 0; PRECISION = 1.0

<u>Query 2.16</u> *Sidewinder hitting an aircraft*

%PARSE_STATS = 2.834 sec.
%MAX_SCORE = 5.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 102.700 sec.

C = [5.0-264968, 4.0-182712, 4.0-216382, 4.0-257055, 2.0-5824, 2.0-258795, 2.0-251701, 2.0-251703, 2.0-251704, 2.0-251706, 2.0-251707, 2.0-251708, 2.0-251709, 2.0-251710, 2.0-161082, 2.0-182711, 2.0-182713, 2.0-242099, 2.0-256393, 2.0-256394, 2.0-256395, 2.0-252492, 2.0-252494, 2.0-252496, 2.0-255577, 2.0-255578, 2.0-255580, 2.0-262868, 2.0-262869, 2.0-262870, 2.0-262871, 2.0-262872, 2.0-262873, 2.0-221353, 2.0-221354, 2.0-3204]

RECALL = 1/4 = 0.25; FALLOUT = 0; PRECISION = 1.0

<u>Query 2.17</u> *a view of a Sidewinder*

%PARSE_STATS = 3.434 sec.
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 72.604 sec.

C = [3.0-29773, 3.0-161082, 3.0-182711, 3.0-182712, 3.0-182713, 3.0-256394, 3.0-256395, 3.0-252492, 3.0-255577, 3.0-255578, 3.0-255580, 3.0-262865, 3.0-262866, 3.0-262867, 3.0-262868, 3.0-262869, 3.0-262870, 3.0-221353, 3.0-221354, 3.0-5824, 2.0-258795, 2.0-251701, 2.0-251703, 2.0-251704, 2.0-251706, 2.0-251707, 2.0-251708, 2.0-251709, 2.0-251710, 2.0-256393, 2.0-216382, 2.0-262871, 2.0-262872, 2.0-262873]

RECALL = 20/20 = 1.0; FALLOUT = 0; PRECISION = 1.0

<u>Query 2.18</u> *Sidewinder at NAF*

%PARSE_STATS = 2.417 sec.

```
%MAX_SCORE = 3.0
%KEYWORD_SCORE = 2.0
%SEARCH_STATS = 46.829 sec.

C = [3.0-256394, 3.0-256395, 3.0-258795, 3.0-262865, 3.0-262866, 3.0-
262867,  3.0-262868, 3.0-262869, 3.0-262870, 3.0-262871, 3.0-262872, 3.0-
262873, 3.0-29773, 3.0-3204, 3.0-256393]

RECALL = 15/15 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. This query requires examining both the parsed caption and the registration data (specifically the *location* attribute) in the *VISUAL* relation. The SQL command inferred by the query structure is:

```
select id
from visual
where location = "Armitage Field"
```

(NAF is being mapped to Armitage Field in the lexicon)

These identifiers are then intersected with the identifiers found in the coarse-grain search (keyword only) to derive those identifiers that meet the coarse-grain threshold and are suitable for fine-grain matching.

Query 2.19 *some type of a Sidewinder*

```
%PARSE_STATS = 2.500 sec.
%MAX_SCORE = 1.0
%KEYWORD_SCORE = 1.0
%SEARCH_STATS = 1.271 sec.
```

Same results as Query 2.1

```
RECALL = 42/42 = 1.0; FALLOUT = 0; PRECISION = 1.0
```

Note. There is little difference between this query and that of Query 2.17 when we are dealing with images. In Query 2.17, we have not filtered out "view" when conducting the search. Hence, we are looking explicitly for either "view" or one of its subclasses to appear in the caption.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Prof. Neil C. Rowe, Code CS/Rp Naval Postgraduate School Monterey, California 93943	2
4.	Prof. Vincent Lum, Code CS/Lm Naval Postgraduate School Monterey, California 93943	2
5.	Prof. Mantak Shing, Code CS/Sh Naval Postgraduate School Monterey, California 93943	1
6.	Prof. Dan Boger, Code AS/Bo Naval Postgraduate School Monterey, California 93943	1
7.	Prof. James Eagle, Code OR/Er Naval Postgraduate School Monterey, California 93943	1
8.	Eugene J. Guglielmo, Code C6344 Naval Air Warfare Center Weapons Division China Lake, California 93555	2
9.	Bill Ball, Code C63 Naval Air Warfare Center Weapons Division China Lake, California 93555	1
10.	John Maynard, Code 402 Naval Ocean Systems Center Command and Control Department San Diego, California 92152	1